Computer Science Matters in Maryland

Computer Science Principles Curriculum and Professional Development Endorsement Submission

Dr. Marie desJardins
Associate Dean for Academic Affairs
College of Engineering and Information Technology
University of Maryland, Baltimore County
1000 Hilltop Circle
Baltimore MD 21250

Dr. Jandelyn (Jan) Plane
Senior Lecturer - Dept. of Computer Science
Director - Center for Women in Computing
Associate Director - ACES Honors Program
University of Maryland - College Park

Email: csmattersinmaryland@gmail.com
Voice: 410-455-3967
Fax: 410-455-3559

Submission Contents

Overview of the Curriculum

Lesson Objectives

Syllabi Comparison

- Authorized Syllabus
- Revised December 2016 Syllabus

CS Matters in Maryland Curriculum

Attestation and Post PD Deliverables

Professional Development Plan

Overview of the Curriculum

Overarching Goals:

The CS Matters in Maryland CS Principles AP course incorporates a focus on active, inquiry-based learning. The structure of the course is designed to meet all of the Computer Science Principles learning objectives, to prepare the students for the two Computer Science Principles performance tasks, and to spread out the work on these tasks over the course of the year. The overarching theme of the course is data: the nature and variety of data on the Internet; algorithmic methods for processing and managing data; and ways in which data can be analyzed, visualized, and interpreted to increase human understanding and solve challenging real-world problems.

Description of the Curriculum:

The curriculum is designed to integrate the computational thinking practices and the big ideas throughout the curriculum with certain objectives emphasized in different units.

Unit 0 and Unit 1 are primarily looking at the impact of computing by having students analyze, explain and evaluate different aspects of computing and technology. The sections within these units look at the impact of technology in general and also look from a high level at the process of creating and adapting technology. Therefore unit 1 also has a significant creative component in order to give motivation for the topics in the later units.

Unit 2 introduces many of the programming aspects so that the skills developed here can be applied in later units and compare using programming tools to answer a question to applying other tools. The basic programming constructs of variables, conditions and loops are introduces as well as more advanced concepts such as abstraction and algorithms.

Unit 3 contributes the data from the Internet by both looking at extracting data and analyzing it, and through those processes introduces the concept of an algorithm. In this unit and in the practice explore task completed, they learn to analyze a problem and artifact .

After Unit 3, the explore task is done to tie in data collected from the internet with the tools to process and analyze it and then effectively communicate their ideas. A practice explore task was done applying a subset of the tools and skills during Unit 1. This explore task encourages the students to use all skills from the first three units to be creative through analyzing the effects of computation.

Unit 4 combines the skills of the previous units by using algorithms and the programming language to develop models and simulations in order to create data that can then be analyzed. This section also especially emphasizes the need to collaborate and work effectively as a team – addressing team structure and evaluation. In addition to the team aspect, the students also learn about the importance of applying modeling and abstraction to problem solving.

Unit 5 continues to use tools that emphasize the manipulation of the data including comparing analysis of the same data done using different tools. Through this manipulation they learn to create a synthesis of algorithms, programming, and data.

Unit 6 brings this section together by introducing new ways to visualize the data and the analysis or manipulation of the data.

The create task is done to allow students to select the tool introduced earlier that allows them to create, analyze and/or visualize data.

This is a generalization of specific information about which big ideas are found to what degree in which units. The details of that section by section analysis is available here: http://csmatters.org/curriculum/summary/objectives/completion

Description of Required Computational Tools:

- Python (and corresponding IDE – e.g. PyCharm)
- Internet and Browser
- Word processor
- Spreadsheet
- On line program development environment such as xx
- Optional units include: Earsketch

Unit Descriptions:

The CS Matters CSP curriculum was developed with one key goal: to provide all students the opportunity to learn computer science within a rigorous and engaging framework. To reach, retain, and teach traditionally underrepresented groups, the curriculum is designed to foster welcoming learning environments that are respectful of the diverse strengths of all students. The theme of the CS Matters course is data -- where it comes from, how it is collected and made available, how it can be analyzed and visualized, and the impact of "big data" on society.

Unit 1, Your Virtual World, informs and involves students in the many ways in which computing shapes their environment. Students study scalable problem solving by participating in citizen science, contributing their thoughts and recording their reactions in daily journals, investigating innovations of particular importance to them, and collaborating with partners and groups. Core lessons from Unit 1 include Impact of Innovation, A Problem Solving Process that Scales, Societal Impact, and Privacy in the Age of Big Data.

Unit 2, Developing Programming Tools, introduces students to software development using the Python programming language. The unit begins by focusing on the motivation for programming and then teaches the fundamentals of procedural programming, including data storage and retrieval, sequence, selection, iteration, and functions. This unit plays a pivotal role that allows subsequent units to challenge students to implement their own code to investigate their virtual world.

Unit 3, Information and the Internet, continues the emphasis on impact while examining the Internet, its core technologies, and its design. The unit explores how the design and technologies of the Internet affect innovation. Students take on the roles of Internet technologies by acting out the parts these technologies play. The first and third units together equip students to complete the College Board's Explore Performance Task.

In the Explore Performance Task (EPT), students choose and explore a computing innovation. The EPT requires students to select and investigate a computational innovation that: has or has had the potential to have significant beneficial and harmful effects on our society, economy, or culture, consumes, produces, and/or transforms data and raises at least one data storage concern, data privacy concern, or data security concern. Students will have 8 hours (480 minutes) of class time to complete the EPT.

Unit 4, Data Acquisition, focuses on data, modeling, and simulations, while introducing fundamental concepts of probability and statistics. Students address the potential and limits of modeling by developing and testing hypotheses. Using computational thinking and the programming skills they learned in the prior unit, students build and test a model that leverages the power of computing to increase the accuracy of its results.

Unit 5, Data Manipulation, orients students to the conceptual foundations and core strategies for managing big data. Students investigate several data manipulation strategies, focusing on common algorithms and methods of evaluating them. The study of algorithms leads to small individual programming projects that acquaint students with the College Board's Create performance task.

Unit 6, Data Visualization, serves as a bridge between the introduction to computing and the development of more substantial programming artifacts. This unit includes several options for teachers to strengthen their students' creative programming abilities. The first lessons use EarSketch to engage students in computation through collaborative music composition. Other lessons use Bokeh from Continuum Analytics to equip students to create their own data visualizations.

In the Create Performance Task (CPT), students bring ideas to life through software development. The CPT requires students to design and iteratively develop a program. The College Board encourages but does not require students to collaborate with a partner, and once students begin the development of the CPT, they may receive help only from the collaborative partner. While not requiring it, this lesson supports a collaborative effort. Even if collaboration is used, each student must independently complete a significant level of planning, designing, and developing the program.

Students will have 12 hours (720 minutes) of class time to complete the CPT. Instruction in the Six Computational Practices and Seven Big Ideas is integrated throughout the course. The following table indicates the number of times each learning objective is addressed in each unit. Each Big Idea is taught in at least three units. Most Big Ideas are taught in every unit.

Major projects and Major activities:

- The Explore Task and its corresponding Practice Explore Task
- The Create Task and its corresponding Practice Create Task

Computational thinking practice and Big Ideas Alignment

Instruction in the Six Computational Practices and Seven Big Ideas is integrated throughout the course. The following table indicates the number of times each learning objective is addressed in each unit. Each Big Idea is taught in at least three units; however, most Big Ideas are taught in every unit. The table below indicates the number of times lesson objectives associated with each big idea are taught in each unit. These are in addition to the instructional values of the Performance Tasks.

Summary of Big Ideas Taught in Each Unit

	Big Ideas						
Unit	1	2	3	4	5	6	7
1	2	2	1	3	0	0	9
2	11	1	1	3	31	0	2
3	11	11	3	7	1	20	8
4	6	9	20	2	9	0	0
5	3	6	7	9	0	1	0
6	27	20	4	6	33	4	4

Six Computational Thinking Practices are used by students in each unit of the course.

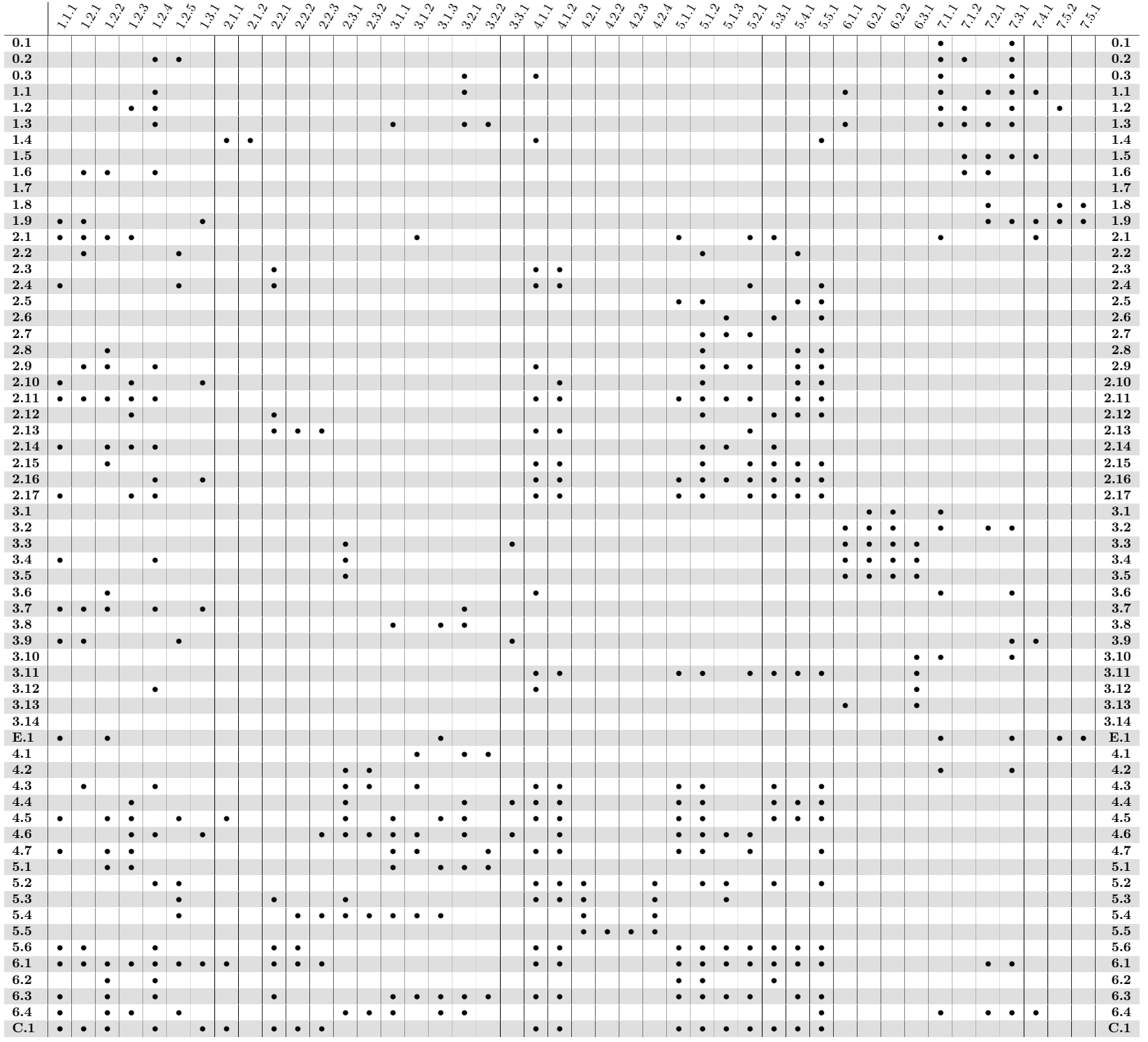
P1: Connecting Computing
P2: Creating Computational Artifacts
P3: Abstracting
P4: Analyzing Problems and Artifacts
P5: Communicating
P6: Collaborating

Seven Big Ideas guide student activities throughout the course.

BI1: Creativity
BI2: Abstraction
BI3: Data and Information
BI4: Algorithms
BI5: Programming
BI6: The Internet
BI7: Global Impact

Lesson Objectives

The curriculum was designed to cover the learning objectives throughout the school year. The first chart displays the lesson objective coverage, and the following table lists the objectives covered in each lesson. The assessment lessons do not list the objectives, but the previous unit objectives are being assessed.


1.1.1 1.2.1 1.2.2 1.2.3 1.2.4 1.2.5 1.3.1 2.1.1 2.1.2 2.2.1 2.2.2 2.2.3 2.3.1 2.3.2 3.1.1 3.1.2 3.1.3 3.2.1 3.2.2 3.3.1 4.1.1 4.1.2 4.2.1 4.2.2 4.2.3 4.2.4 5.1.1 5.1.2 5.1.3 5.2.1 5.3.1 5.4.1 5.5.1 6.1.1 6.2.1 6.2.2 6.3.1 7.1.1 7.1.2 7.2.1 7.3.1 7.4.1 7.5.2 7.5.1
0.1 0.2 0.3 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14 2.15 2.16 2.17 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 E.1 4.1 4.2 4.3 4.4 4.5 4.6 4.7 5.1 5.2 5.3 5.4 5.5 5.6 6.1 6.2 6.3 6.4 C.1

Lesson	Objectives
0.1	7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.3.1 Analyze the beneficial and harmful effects of computing.
0.2	1.2.4 Collaborate in the creation of computational artifacts. 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.1.2 Explain how people participate in a problem solving process that scales. 7.3.1 Analyze the beneficial and harmful effects of computing.
0.3	3.2.1 Extract information from data to discover and explain connections or trends 4.1.1 Develop an algorithm for implementation in a program. 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.3.1 Analyze the beneficial and harmful effects of computing.
1.1	1.2.4 Collaborate in the creation of computational artifacts. 3.2.1 Extract information from data to discover and explain connections or trends 6.1.1 Explain the abstractions in the Internet and how the Internet functions. 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.2.1 Explain how computing has impacted innovations in other fields. 7.3.1 Analyze the beneficial and harmful effects of computing. 7.4.1 Explain the connections between computing and real-world contexts, including economic, social,...
1.2	1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 1.2.4 Collaborate in the creation of computational artifacts. 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.1.2 Explain how people participate in a problem solving process that scales. 7.3.1 Analyze the beneficial and harmful effects of computing. 7.5.2 Evaluate online and print sources for appropriateness and credibility.
1.3	1.2.4 Collaborate in the creation of computational artifacts. 3.1.1 Find patterns and test hypotheses about digitally processed information to gain insight and kn... 3.2.1 Extract information from data to discover and explain connections or trends 3.2.2 Determine how large data sets impact the use of computational processes to discover informatio... 6.1.1 Explain the abstractions in the Internet and how the Internet functions. 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.1.2 Explain how people participate in a problem solving process that scales. 7.2.1 Explain how computing has impacted innovations in other fields. 7.3.1 Analyze the beneficial and harmful effects of computing.
1.4	2.1.1 Describe the variety of abstractions used to represent data. 2.1.2 Explain how binary sequences are used to represent digital data. 4.1.1 Develop an algorithm for implementation in a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
1.5	7.1.2 Explain how people participate in a problem solving process that scales. 7.2.1 Explain how computing has impacted innovations in other fields. 7.3.1 Analyze the beneficial and harmful effects of computing. 7.4.1 Explain the connections between computing and real-world contexts, including economic, social,...
1.6	1.2.1 Create a computational artifact for creative expression. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.4 Collaborate in the creation of computational artifacts. 7.1.2 Explain how people participate in a problem solving process that scales. 7.2.1 Explain how computing has impacted innovations in other fields.
1.7	*Assessment*
1.8	7.2.1 Explain how computing has impacted innovations in other fields. 7.5.2 Evaluate online and print sources for appropriateness and credibility. 7.5.1 Access, manage, and attribute information using effective strategies.
1.9	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.1 Create a computational artifact for creative expression. 1.3.1 Use computing tools and techniques for creative expression. 7.2.1 Explain how computing has impacted innovations in other fields. 7.3.1 Analyze the beneficial and harmful effects of computing. 7.4.1 Explain the connections between computing and real-world contexts, including economic, social,... 7.5.2 Evaluate online and print sources for appropriateness and credibility. 7.5.1 Access, manage, and attribute information using effective strategies.

2.1	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.1 Create a computational artifact for creative expression. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 3.1.2 Collaborate when processing information to gain insight and knowledge. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.2.1 Explain how programs implement algorithms. 5.3.1 Use abstraction to manage complexity in programs. 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.4.1 Explain the connections between computing and real-world contexts, including economic, social,...
2.2	1.2.1 Create a computational artifact for creative expression. 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. 5.1.2 Develop a correct program to solve problems. 5.4.1 Evaluate the correctness of a program.
2.3	2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language.
2.4	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.2.1 Explain how programs implement algorithms. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
2.5	5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
2.6	5.1.3 Collaborate to develop a program. 5.3.1 Use abstraction to manage complexity in programs. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
2.7	5.1.2 Develop a correct program to solve problems. 5.1.3 Collaborate to develop a program. 5.2.1 Explain how programs implement algorithms.
2.8	1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 5.1.2 Develop a correct program to solve problems. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
2.9	1.2.1 Create a computational artifact for creative expression. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.4 Collaborate in the creation of computational artifacts. 4.1.1 Develop an algorithm for implementation in a program. 5.1.2 Develop a correct program to solve problems. 5.1.3 Collaborate to develop a program. 5.2.1 Explain how programs implement algorithms. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
2.10	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 1.3.1 Use computing tools and techniques for creative expression. 4.1.2 Express an algorithm in a language. 5.1.2 Develop a correct program to solve problems. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
2.11	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.1 Create a computational artifact for creative expression. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 1.2.4 Collaborate in the creation of computational artifacts.

	4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.1.3 Collaborate to develop a program. 5.2.1 Explain how programs implement algorithms. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
2.12	1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. 5.1.2 Develop a correct program to solve problems. 5.3.1 Use abstraction to manage complexity in programs. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
2.13	2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. 2.2.2 Use multiple levels of abstraction to write programs. 2.2.3 Identify multiple levels of abstractions used when writing programs. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.2.1 Explain how programs implement algorithms.
2.14	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 1.2.4 Collaborate in the creation of computational artifacts. 5.1.2 Develop a correct program to solve problems. 5.1.3 Collaborate to develop a program. 5.3.1 Use abstraction to manage complexity in programs.
2.15	1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.2 Develop a correct program to solve problems. 5.2.1 Explain how programs implement algorithms. 5.3.1 Use abstraction to manage complexity in programs. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
2.16	1.2.4 Collaborate in the creation of computational artifacts. 1.3.1 Use computing tools and techniques for creative expression. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.1.3 Collaborate to develop a program. 5.2.1 Explain how programs implement algorithms. 5.3.1 Use abstraction to manage complexity in programs. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
2.17	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 1.2.4 Collaborate in the creation of computational artifacts. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.2.1 Explain how programs implement algorithms. 5.3.1 Use abstraction to manage complexity in programs. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
3.1	6.2.1 Explain characteristics of the Internet and the systems built on it. 6.2.2 Explain how the characteristics of the Internet influence the systems built on it.

	7.1.1 Explain how computing innovations affect communication, interaction and cognition.
3.2	6.1.1 Explain the abstractions in the Internet and how the Internet functions. 6.2.1 Explain characteristics of the Internet and the systems built on it. 6.2.2 Explain how the characteristics of the Internet influence the systems built on it. 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.2.1 Explain how computing has impacted innovations in other fields. 7.3.1 Analyze the beneficial and harmful effects of computing.
3.3	2.3.1 Use models and simulations to represent phenomena. 3.3.1 Analyze how data representation, storage, security, and transmission of data involve computati... 6.1.1 Explain the abstractions in the Internet and how the Internet functions. 6.2.1 Explain characteristics of the Internet and the systems built on it. 6.2.2 Explain how the characteristics of the Internet influence the systems built on it. 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with th...
3.4	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.4 Collaborate in the creation of computational artifacts. 2.3.1 Use models and simulations to represent phenomena. 6.1.1 Explain the abstractions in the Internet and how the Internet functions. 6.2.1 Explain characteristics of the Internet and the systems built on it. 6.2.2 Explain how the characteristics of the Internet influence the systems built on it. 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with th...
3.5	2.3.1 Use models and simulations to represent phenomena. 6.1.1 Explain the abstractions in the Internet and how the Internet functions. 6.2.1 Explain characteristics of the Internet and the systems built on it. 6.2.2 Explain how the characteristics of the Internet influence the systems built on it. 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with th...
3.6	1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 4.1.1 Develop an algorithm for implementation in a program. 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.3.1 Analyze the beneficial and harmful effects of computing.
3.7	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.1 Create a computational artifact for creative expression. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.4 Collaborate in the creation of computational artifacts. 1.3.1 Use computing tools and techniques for creative expression. 3.2.1 Extract information from data to discover and explain connections or trends
3.8	3.1.1 Find patterns and test hypotheses about digitally processed information to gain insight and kn... 3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate vi... 3.2.1 Extract information from data to discover and explain connections or trends
3.9	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.1 Create a computational artifact for creative expression. 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. 3.3.1 Analyze how data representation, storage, security, and transmission of data involve computati... 7.3.1 Analyze the beneficial and harmful effects of computing. 7.4.1 Explain the connections between computing and real-world contexts, including economic, social,...
3.10	6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with th... 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.3.1 Analyze the beneficial and harmful effects of computing.
3.11	4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.2.1 Explain how programs implement algorithms. 5.3.1 Use abstraction to manage complexity in programs. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming. 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with th...
3.12	1.2.4 Collaborate in the creation of computational artifacts. 4.1.1 Develop an algorithm for implementation in a program. 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with th...

3.13	6.1.1 Explain the abstractions in the Internet and how the Internet functions. 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with th...
3.14	*Assessment*
E.1	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate vi... 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.3.1 Analyze the beneficial and harmful effects of computing. 7.5.2 Evaluate online and print sources for appropriateness and credibility. 7.5.1 Access, manage, and attribute information using effective strategies.
4.1	3.1.2 Collaborate when processing information to gain insight and knowledge. 3.2.1 Extract information from data to discover and explain connections or trends 3.2.2 Determine how large data sets impact the use of computational processes to discover informatio...
4.2	2.3.1 Use models and simulations to represent phenomena. 2.3.2 Use models and simulations to formulate, refine and test hypotheses. 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.3.1 Analyze the beneficial and harmful effects of computing.
4.3	1.2.1 Create a computational artifact for creative expression. 1.2.4 Collaborate in the creation of computational artifacts. 2.3.1 Use models and simulations to represent phenomena. 2.3.2 Use models and simulations to formulate, refine and test hypotheses. 3.1.2 Collaborate when processing information to gain insight and knowledge. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.3.1 Use abstraction to manage complexity in programs. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
4.4	1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 2.3.1 Use models and simulations to represent phenomena. 3.2.1 Extract information from data to discover and explain connections or trends 3.3.1 Analyze how data representation, storage, security, and transmission of data involve computati... 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.3.1 Use abstraction to manage complexity in programs. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
4.5	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. 2.1.1 Describe the variety of abstractions used to represent data. 2.3.1 Use models and simulations to represent phenomena. 3.1.1 Find patterns and test hypotheses about digitally processed information to gain insight and kn... 3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate vi... 3.2.1 Extract information from data to discover and explain connections or trends 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.3.1 Use abstraction to manage complexity in programs. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
4.6	1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 1.2.4 Collaborate in the creation of computational artifacts. 1.3.1 Use computing tools and techniques for creative expression. 2.2.3 Identify multiple levels of abstractions used when writing programs.

	2.3.1 Use models and simulations to represent phenomena. 2.3.2 Use models and simulations to formulate, refine and test hypotheses. 3.1.1 Find patterns and test hypotheses about digitally processed information to gain insight and kn... 3.1.2 Collaborate when processing information to gain insight and knowledge. 3.2.1 Extract information from data to discover and explain connections or trends 3.3.1 Analyze how data representation, storage, security, and transmission of data involve computati... 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.1.3 Collaborate to develop a program. 5.2.1 Explain how programs implement algorithms.
4.7	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 3.1.1 Find patterns and test hypotheses about digitally processed information to gain insight and kn... 3.1.2 Collaborate when processing information to gain insight and knowledge. 3.2.2 Determine how large data sets impact the use of computational processes to discover informatio... 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.2.1 Explain how programs implement algorithms. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
5.1	1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 3.1.1 Find patterns and test hypotheses about digitally processed information to gain insight and kn... 3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate vi... 3.2.1 Extract information from data to discover and explain connections or trends 3.2.2 Determine how large data sets impact the use of computational processes to discover informatio...
5.2	1.2.4 Collaborate in the creation of computational artifacts. 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 4.2.1 Explain the difference between algorithms that run in a reasonable time and those that do not ... 4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness and clarity. 5.1.2 Develop a correct program to solve problems. 5.1.3 Collaborate to develop a program. 5.3.1 Use abstraction to manage complexity in programs. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
5.3	1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. 2.3.1 Use models and simulations to represent phenomena. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 4.2.1 Explain the difference between algorithms that run in a reasonable time and those that do not ... 4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness and clarity. 5.1.3 Collaborate to develop a program.
5.4	1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. 2.2.2 Use multiple levels of abstraction to write programs. 2.2.3 Identify multiple levels of abstractions used when writing programs. 2.3.1 Use models and simulations to represent phenomena. 2.3.2 Use models and simulations to formulate, refine and test hypotheses. 3.1.1 Find patterns and test hypotheses about digitally processed information to gain insight and kn... 3.1.2 Collaborate when processing information to gain insight and knowledge. 3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate vi... 4.2.1 Explain the difference between algorithms that run in a reasonable time and those that do not ... 4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness and clarity.
5.5	4.2.1 Explain the difference between algorithms that run in a reasonable time and those that do not ... 4.2.2 Explain the difference between solvable and unsolvable problems in computer science. 4.2.3 Explain the existence of undecidable problems in computer science.

	4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness and clarity.
5.6	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.1 Create a computational artifact for creative expression. 1.2.4 Collaborate in the creation of computational artifacts. 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. 2.2.2 Use multiple levels of abstraction to write programs. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.1.3 Collaborate to develop a program. 5.2.1 Explain how programs implement algorithms. 5.3.1 Use abstraction to manage complexity in programs. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
6.1	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.1 Create a computational artifact for creative expression. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 1.2.4 Collaborate in the creation of computational artifacts. 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. 1.3.1 Use computing tools and techniques for creative expression. 2.1.1 Describe the variety of abstractions used to represent data. 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. 2.2.2 Use multiple levels of abstraction to write programs. 2.2.3 Identify multiple levels of abstractions used when writing programs. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.1.3 Collaborate to develop a program. 5.2.1 Explain how programs implement algorithms. 5.3.1 Use abstraction to manage complexity in programs. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming. 7.2.1 Explain how computing has impacted innovations in other fields. 7.3.1 Analyze the beneficial and harmful effects of computing.
6.2	1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.4 Collaborate in the creation of computational artifacts. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.3.1 Use abstraction to manage complexity in programs.
6.3	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.4 Collaborate in the creation of computational artifacts. 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. 3.1.1 Find patterns and test hypotheses about digitally processed information to gain insight and kn... 3.1.2 Collaborate when processing information to gain insight and knowledge. 3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate vi... 3.2.1 Extract information from data to discover and explain connections or trends 3.2.2 Determine how large data sets impact the use of computational processes to discover informatio... 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.1.3 Collaborate to develop a program. 5.2.1 Explain how programs implement algorithms. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.
6.4	1.1.1 Apply a creative development process when creating computational artifacts.

	1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. 2.3.1 Use models and simulations to represent phenomena. 2.3.2 Use models and simulations to formulate, refine and test hypotheses. 3.1.1 Find patterns and test hypotheses about digitally processed information to gain insight and kn... 3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate vi... 3.2.1 Extract information from data to discover and explain connections or trends 5.5.1 Employ appropriate mathematical and logical concepts in programming. 7.1.1 Explain how computing innovations affect communication, interaction and cognition. 7.2.1 Explain how computing has impacted innovations in other fields. 7.3.1 Analyze the beneficial and harmful effects of computing. 7.4.1 Explain the connections between computing and real-world contexts, including economic, social,...
C.1	1.1.1 Apply a creative development process when creating computational artifacts. 1.2.1 Create a computational artifact for creative expression. 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. 1.2.4 Collaborate in the creation of computational artifacts. 1.3.1 Use computing tools and techniques for creative expression. 2.1.1 Describe the variety of abstractions used to represent data. 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. 2.2.2 Use multiple levels of abstraction to write programs. 2.2.3 Identify multiple levels of abstractions used when writing programs. 4.1.1 Develop an algorithm for implementation in a program. 4.1.2 Express an algorithm in a language. 5.1.1 Develop a program for creative expression to satisfy personal curiosity or to create new knowl... 5.1.2 Develop a correct program to solve problems. 5.1.3 Collaborate to develop a program. 5.2.1 Explain how programs implement algorithms. 5.3.1 Use abstraction to manage complexity in programs. 5.4.1 Evaluate the correctness of a program. 5.5.1 Employ appropriate mathematical and logical concepts in programming.

Authorized Syllabus Curriculum Comparison to Current Version

The CS Matters syllabus was originally authorized by the College Board in August 2016. Since that time, we have updated and improved aspects of the curriculum, and included a correspondingly updated syllabus in this submission. This section compares the College Board approved syllabus to the updated December 2016 syllabus. Both syllabi are included in this endorsement submission

1. The syllabus begins with a textual overview of the curriculum. The overview addresses the theme of each unit in the order addressed. The overview is consistent with the current published curriculum. Following the overview, there is a frequency table showing the number of times each big idea is addressed each unit.

.

2. Each unit begins with a brief description of the unit. These are consistent with the current curriculum.

3. After the description is a list of lessons taught in the unit. The table below indicates updates to the lessons that occurred between the August 2016 authorized syllabus and the December 2016 (current) version of the syllabus.

Updates to Lessons List	
Unit Number	Updates
1	none
2	none
3	Lessons designated as optional have been updated. Lesson 3-12, 3-13 and 3-14 have been added.
4	none
5	none
6	Lesson 6-4 was added to the curriculum.

4. Following the lesson list in each unit is a list of College Board Learning Objectives taught by each unit. The table below indicates differences between the coverage of the College Board authorized syllabus and the December 2016 (current) version of the syllabus. Note that although some learning objectives have been rearranged, the curriculum still covers all learning objectives thoroughly, as shown in the mapping.

Updates in Learning Objective Coverage		
Unit Number	Learning Objectives that appear in the December 2016 syllabus but not in the authorized syllabus	Learning Objectives that appear in the authorized syllabus but not the December 2016 syllabus
1	1.2.4, 3.1.1, 3.2.1, 3.2.2, 4.1.1, 7.5.1, 7.5.2	1.2.5, 2.2.3, 3.1.3, 3.3.1, 6.2.1
2	2.2.1, 7.1.1, 7.4.1	2.3.1, 2.3.2, 3.1.3, 3.1 3.2.1, 3.2.2, 3.3.1
3	1.2.4, 3.1.1, 3.1.3, 3.2.1, 5.1.2, 5.2.1, 5.3.1, 5.4.1,5.5.1	1.2.3, 4.2.3
4	2.1.1, 4.1.1, 4.1.2	4.2.1
5	2.1.2, 3.1.1	none
6	1.3.1, 2.1.1, 2.3.1, 3.1.1,3.2.2, 7.1.1, 7.2.1, 7.3.1, 7.4.1	4.2.4, 6.1.1

5. Following the list of College Board Learning Objectives taught in each unit is a sample assignment from each unit. Each assignment description is followed by a list of the Learning Objectives addressed and the computational thinking practice associated with each learning objectives. The sample assignments are the same for both versions of the curriculum, and are consistent with both syllabi. The sample activities in both versions are correctly associated with computational thinking practices.

6. The authorized syllabus and the December 2016 syllabus both indicate the required minimum times allotted for each performance task.

Resubmission of the Syllabus after a Revision

In the College Board's announcement of approval of the CS Matters Syllabus for the Computer Science Principles course for the 2016 - 2017 academic year, the College Board stated:

> You do not need to resubmit the syllabus unless the College Board significantly revises the AP course.
>
> We understand that you may need to modify or adapt your syllabus or course plan to address the needs of your students, reflect new discoveries, and try out new approaches. You do not need to resubmit your syllabus when you modify it, as long as you are not changing your course so significantly as to eliminate from your curriculum one or more of the curricular requirements on the AP Course Audit form.

The updates in December 2016 syllabus do not modify any of the elements required to meet the curricular requirements on the AP Course Audit form. We would appreciate any guidance from the College Board about whether we need to have the new syllabus authorized as part of the endorsement process.

Advanced Placement

Computer Science Principles

Syllabus

Instructor

Teacher:

School:

Course Overview

CS Matters in Maryland (http://csmatters.org/) is an NSF-funded CS10K project focusing on the new AP Computer Science Principles (CSP) course. The objectives of the course are consistent with those defined within the AP Computer Science Principles Curriculum Framework by the College Board. This course will be taught using the curriculum developed by CS Matters.

The CS Matters CSP curriculum was developed with one key goal: to provide all students the opportunity to learn computer science within a rigorous and engaging framework. To reach, retain, and teach traditionally underrepresented groups, the curriculum is designed to foster welcoming learning environments that are respectful of the diverse strengths of all students. The theme of the CS Matters course is *data* -- where it comes from, how it is collected and made available, how it can be analyzed and visualized, and the impact of “big data” on society.

Unit 1, *Your Virtual World*, informs and involves students in the many ways in which computing shapes their environment. Students study scalable problem solving by participating in citizen science, contributing their thoughts and recording their reactions in daily journals, investigating innovations of particular importance to them, and collaborating with partners and groups. Core lessons from Unit 1 include Impact of Innovation, A Problem Solving Process that Scales, Societal Impact, and Privacy in the Age of Big Data.

Unit 2, *Developing Programming Tools*, introduces students to software development using the Python programming language. The unit begins by focusing on the motivation for programming and then teaches the fundamentals of procedural programming, including data storage and retrieval, sequence, selection, iteration, and functions. This unit plays a pivotal role that allows subsequent units to challenge students to implement their own code to investigate their virtual world.

Unit 3, *Information and the Internet*, continues the emphasis on impact while examining the Internet, its core technologies, and its design. The unit explores how the design and technologies of the Internet affect innovation. Students take on the roles of Internet technologies by acting out the parts these technologies play. The first and third units together equip students to complete the College Board's *Explore* Performance Task.

In the Explore Performance Task (EPT), students choose and explore a computing innovation. The EPT requires students to select and investigate a computational innovation that: has or has had the potential to have significant beneficial and harmful effects on our society, economy, or culture, consumes, produces, and/or transforms data and raises at least one data storage concern, data privacy concern, or data security concern. Students will have 8 hours (480 minutes) of class time to complete the EPT.

Unit 4, *Data Acquisition*, focuses on data, modeling, and simulations, while introducing fundamental concepts of probability and statistics. Students address the potential and limits of modeling by developing and testing hypotheses. Using computational thinking and the programming skills they learned in the prior unit, students build and test a model that leverages the power of computing to increase the accuracy of its results.

Unit 5, *Data Manipulation*, orients students to the conceptual foundations and core strategies for managing big data. Students investigate several data manipulation strategies, focusing on common algorithms and methods of evaluating them. The study of algorithms leads to small individual programming projects that acquaint students with the College Board's *Create* performance task.

Unit 6, *Data Visualization*, serves as a bridge between the introduction to computing and the development of more substantial programming artifacts. This unit includes several options for teachers to strengthen their students' creative programming abilities. The first lessons use EarSketch to engage students in computation through collaborative music composition. Other lessons use Bokeh from Continuum Analytics to equip students to create their own data visualizations.

In the Create Performance Task (CPT), students bring ideas to life through software development. The CPT requires students to design and iteratively develop a program. The College Board encourages but does not require students to collaborate with a partner, and once students begin the development of the CPT, they may receive help only from the collaborative partner. While not requiring it, this lesson supports a collaborative effort. Even if collaboration is used, each student must independently complete a significant level of planning, designing, and developing the program. Students will have 12 hours (720 minutes) of class time to complete the CPT.

Instruction in the Six Computational Practices and Seven Big Ideas is integrated throughout the course. The following table indicates the number of times each learning objective is addressed in each unit. Each Big Idea is taught in at least three units. Most

Big Ideas are taught in every unit. The table below indicates the number of times lesson objectives associated with each big idea are taught in each unit. These are in addition to the instructional values of the Performance Tasks.

Summary of Big Ideas Taught in Each Unit							
	Big Ideas						
Unit	1	2	3	4	5	6	7
1	2	2	1	3	0	0	9
2	11	1	1	3	31	0	2
3	11	11	3	7	1	20	8
4	6	9	20	2	9	0	0
5	3	6	7	9	0	1	0
6	27	20	4	6	33	4	4

Six Computational Thinking Practices are used by students in each unit of the course.

P1: Connecting Computing
P2: Creating Computational Artifacts
P3: Abstracting
P4: Analyzing Problems and Artifacts
P5: Communicating
P6: Collaborating

Seven Big Ideas guide student activities throughout the course.

BI1: Creativity
BI2: Abstraction
BI3: Data and Information
BI4: Algorithms
BI5: Programming
BI6: The Internet
BI7: Global Impact

Sample Assignment Associations			
Sample Assignments	Page Numbers	Computational Practices	Big Ideas
1	5	1, 4 and 6	1, 6 and 7
2	7	2, 4 and 6	1, 4 and 5
3	11	1, 2, 3, 4 and 5	2 and 6
4	14	1, 3 and 5	2, 3 and 7
5	17	1, 4 and 5	1 and 3
6	20	2, 3, 5 and 6	1, 2, 4 and 5

Course Content by Unit

1. Your Virtual World

The main idea of Unit 1 is to explore the effect of data on students' lives. Students will discover what is known about them online, explore the issue of data privacy, and the rights to access or change that data. After finding ways that the world of information is changing, students will learn how data is stored using binary codes, hardware and files.

Unit 1 Lessons:

1 - 1. Into the Darkness: A World Without Digital Communication
1 - 2. Into the Light: How Computers and the Internet Enhance Innovation
1 - 3. Exploring Innovations
1 - 4. It's Just Bits
1 - 5. How Innovation Affects Our Lives
1 - 6. A Problem Solving Process that Scales
1 - 7. Unit 1 Assessment
1 - 8. Practice Performance Task

Unit 1 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.3**	**Create a new computational artifact by combining or modifying existing artifacts.**
Creativity	**1.2.5**	**Analyze the correctness, usability, functionality, and suitability of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.1.1**	**Describe the variety of abstractions used to represent data.**
Abstraction	**2.1.2**	**Explain how binary sequences are used to represent digital data.**
Abstraction	**2.2.3**	**Identify multiple levels of abstractions used when writing programs.**
Data	**3.1.3**	**Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.**
Data	**3.3.1**	**Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information.**
Internet	**6.1.1**	**Explain the abstractions in the Internet and how it functions.**
Internet	**6.2.1**	**Explain characteristics of the Internet and the systems built on it.**
Internet	**6.2.2**	**Explain how the characteristics of the Internet influence the systems built on it.**

Impact	**7.1.1**	**Explain how computing innovations affect communication, interaction and cognition.**
Impact	**7.1.2**	**Explain how people participate in a problem solving process that scales.**
Impact	**7.2.1**	**Explain how computing has impacted innovations in other fields.**
Impact	**7.3.1**	**Analyze the beneficial and harmful effects of computing.**
Impact	**7.4.1**	**Explain the connections between computing and economic, social and cultural contexts.**

Computational Thinking Practices used in Unit 1: P1, P2, P4, P5 & P6

Sample Assignment 1

Students work in pairs to examine Wikipedia article, “Ten Commandments of Computer Ethics” (http://en.wikipedia.org/wiki/Ten_Commandments_of_Computer_Ethics) and identify and share the two commandments they think are the most commonly violated.

Pairs discuss how the commandments can be violated using cloud technology or social media.

Pairs prepare to share with the class how two rules are commonly violated. Identify the commandments commonly violated and the impact or consequence of each violation.

As a class, produce a revised version of the commandments.

Associated Computational Practices: P1, P4 and P6

Associated Big Ideas: BI1, BI6, BI7

Learning Objectives and Associated Computational Practices:

LO 1.2.4 [P6]

LO 6.3.1 [P1]

LO 7.1.1, [P4]

LO 7.3.1 [P4]

LO 7.4.1 [P1]

2. Developing Programs

The main idea of Unit 2 is to create solutions with code, debug, and verify results. Programming uses creative expression to solve problems correctly using abstraction, both individually and collaboratively. Students implement algorithms using math and logic, and evaluate programs for correctness.

The programming unit takes the formality of algorithm expression one step further by having students write programs in Python. In the programming unit, the levels of abstraction increase as students learn how a programming language can be used to control the machine on which the program is running. The topics of the programming unit are input/output, calculations with numbers, branching statements and Boolean logic, iteration, procedural abstraction, and processing data in a list.

We plan to use Python 3.x with the PyCharm IDE and the textbook *How to Think Like a Computer Scientist,* hosted by Runestone Interactive: (http://interactivepython.org/runestone/static/CCPS_Python/index.html)

Unit 2 Lessons:

2 - 1. Programming: Introduction and Motivation
2 - 2. Using Python and PyCharm
2 - 3. Algorithms: Basics
2 - 4. Algorithms: Pseudocode
2 - 5. Reading Python Code and Debugging
2 - 6. Types and Evaluation
2 - 7. Creating and Assigning Variables
2 - 8. Comparison, Logical Operators, and Conditional Execution
2 - 9. Nested and Chained Conditional Statements
2 - 10. Iteration: For Loops
2 - 11. Iteration: While Loops
2 - 12. Functions: Parameters and Return Values
2 - 13. Algorithms: Layers of Abstraction
2 - 14. Functions: Scope and Abstraction
2 - 15. Strings: Traversing, Slicing, and Parsing
2 - 16. Lists: Creation, Traversal, Insertion, and Removal
2 - 17. Unit Assessment

Unit 2 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.3**	**Create a new computational artifact by combining or modifying existing artifacts.**

Creativity	**1.2.4**	**Collaborate in the creation of computational artifacts.**
Creativity	**1.2.5**	**Analyze the correctness, usability, functionality, and suitability of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.1.2**	**Explain how binary sequences are used to represent digital data.**
Abstraction	**2.2.3**	**Identify multiple levels of abstractions used when writing programs.**
Abstraction	**2.3.1**	**Use models and simulations to represent phenomena.**
Abstraction	**2.3.2**	**Use models and simulations to formulate, refine and test hypothesis.**
Data	**3.1.1**	**Use computers to process information, find patterns and test hypothesis about digitally processed information to gain insight and knowledge.**
Data	**3.1.2**	**Collaborate when processing information to gain insight and knowledge.**
Data	**3.1.3**	**Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.**
Data	**3.2.1**	**Extract information from data to discover and explain connections, patterns or trends.**
Data	**3.2.2**	**Use large data sets to explore and discover information and knowledge.**
Data	**3.3.1**	**Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information.**
Algorithms	**4.1.1**	**Develop an algorithm for implementation in a program.**
Algorithms	**4.1.2**	**Express an algorithm in a language.**
Programming	**5.1.2**	**Develop a correct program to solve problems.**
Programming	**5.1.3**	**Collaborate to develop a program.**
Programming	**5.2.1**	**Explain how programs implement algorithms.**
Programming	**5.3.1**	**Use abstraction to manage complexity in programs.**
Programming	**5.4.1**	**Evaluate the correctness of a program.**
Programming	**5.5.1**	**Employ appropriate mathematical and logical concepts in programming.**

Computational Thinking Practices used in Unit 2: P1, P2, P3, P4, P5 and P6

Sample Assignment 2

LinkyListy Role Play:

Create a human “list of students” by starting with an empty list the front of the room and then modifying the list by following the following program.

Designate a section of the board or a poster to act as the console/printer for output and a volunteer student to act as the printer driver.

Designate an area at the front of the room for the computer memory (and future students).

The names should be changed to match those of students in the class. Also designate three students to play the role of fish and one as a dog.

As students are called by the program, have each student come up to stand in front of the room where memory resides. Have each student point to the student that follows them to form a list as they come to the front.

```
students = []
students.append('Joe')
students.append('Pat')
students.append('Alea')
students.append('Marta')
print(students)
#add additional commands to append or insert  4 or 5 more
students
print(students)
print(len(students))
print(students[3])
students.reverse()
print(students)
print('David' in students)
students.sort()
print(students)
more = ['Tom', 'Laverne']
students = students + more
print(students)
pets = ['fish'*3,'dog']
del students[1]
del students[2:4]
students.insert(0, 'Jennifer')
students = students + pets
print(students)
print(students.index('Marta'))
```

Associated Computational Practices: P2, P3, P4, P6

Associated Big Ideas: BI1, BI4, BI5

Learning Objectives and Associated Computational Practices:

LO 1.2.4 [P6]

LO 4.1.1 [P2]

LO 5.1. [P6]

LO 5.3.1 [P3]

LO 5.4.1 [P4]

3. Information and the Internet

The main idea of Unit 3 is to explore the Internet to prepare students to do research on a current Internet related topic in detail on their own. A variety of algorithms from different areas of interest are investigated. Students will learn how information is transmitted online and how search engines find and organize the data using complex algorithms. Studying cybersecurity and cryptography allows students to understand more about the issues concerning the privacy of data online which leads to a discussion on the ethics and social issues affecting the increased use of online data.

Unit 3 Lessons:

3 - 1. The Internet: Basics of Information Transmission
3 - 2. The Internet: Present and Future
3 - 3. How the Internet Works: Routing
Optional 3 - 4. How the Internet Works: Domain Name System
Optional 3 - 5. How the Internet Works: DNS Activity
3 - 6. Search Engines: Finding Information
3 - 7. Search Engines: Page Rank and Retrieval
3 - 8. Basic Statistics with Excel
3 - 9. Practice for Explore Performance Task
3 - 10. Cybersecurity: Attacks, Protection, and Impact
3 - 11. Cryptography: Symmetric Encryption

Unit 3 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.3**	**Create a new computational artifact by combining or modifying existing artifacts.**
Creativity	**1.2.5**	**Analyze the correctness, usability, functionality, and suitability of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.1.1**	**Describe the variety of abstractions used to represent data.**
Abstraction	**2.1.2**	**Explain how binary sequences are used to represent digital data.**
Data	**3.3.1**	**Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information.**
Algorithms	**4.1.1**	**Develop an algorithm for implementation in a program.**
Algorithms	**4.2.2**	**Explain the difference between solvable and unsolvable problems in computer science.**
Algorithms	**4.2.3**	**Explain the existence of undecidable problems in computer science.**
Programming	**5.1.1**	**Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.**

Internet	**6.1.1**	**Explain the abstractions in the Internet and how it functions.**
Internet	**6.2.1**	**Explain characteristics of the Internet and the systems built on it.**
Internet	**6.2.2**	**Explain how the characteristics of the Internet influence the systems built on it.**
Internet	**6.3.1**	**Identify existing cybersecurity concerns and potential options that address these issues with the Internet and the systems built on it.**
Impact	**7.1.1**	**Explain how computing innovations affect communication, interaction and cognition.**
Impact	**7.3.1**	**Analyze the beneficial and harmful effects of computing.**
Impact	**7.4.1**	**Explain the connections between computing and economic, social and cultural contexts.**

Computational Thinking Practices used in Unit 3: P1, P2, P3, P4, P5 and P6

Sample Assignment 3

In this lesson, students will expand their knowledge of how the Domain Name System (DNS) works by acting as a class to simulate the use of DNS to retrieve web pages.

Once the simulation is functioning students enhance its efficiency through the use of caching.

Poison the DNS cache by adding false DNS replies (DNS poisoning).

Students discuss with their groups how DNS works and how it supports Internet growth. Then they explain in their journals how:

- DNS works.
- Caching is both a benefit and a security risk.
- DNS supports Internet growth.

Associated Computational Practices: P1, P2, P3, P4, P5

Associated Big Ideas: BI2, BI6

Learning Objectives and Associated Computational Practices:

LO 2.3.1 [P3]

LO 4.1.1 [P2]

LO 6.1,1 [P3]

LO 6.2.1 [P5]

LO 6.2.2 [P4]

LO 6.3.1 [P1]

Performance Task: **Explore**. Research an innovation that uses the Internet.

Students will have 8 hours (480 minutes) of class time to complete this performance task.

4. Data Acquisition

The main idea of Unit 4 is to enable students to find data online and analyze it using a spreadsheet and Python code. To prepare for the performance task, students will follow the pattern of developing a hypothesis and then digging into data to find the answer. File input and output will allow students to use large sets of data, find patterns, look for metadata (data about data) and create visualizations of the data. Students will also investigate what is a reasonable solution and how the new data revolution can drive discovery and decision making.

Unit 4 Lessons:

4 - 1. Data Acquisition and Analysis
4 - 2. What are Models and Simulations?
4 - 3. Using Data and Simulations
4 - 4. File Input and Output using Python
4 - 5. Data Collection, Analysis, and Simulation
Optional 4 - 6. Hypothesis Testing with Simulations in NetLogo
4 - 7. Unit 4 Assessment

Unit 4 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.3**	**Create a new computational artifact by combining or modifying existing artifacts.**
Creativity	**1.2.4**	**Collaborate in the creation of computational artifacts.**
Creativity	**1.2.5**	**Analyze the correctness, usability, functionality, and suitability of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.2.3**	**Identify multiple levels of abstractions used when writing programs.**
Abstraction	**2.3.1**	**Use models and simulations to represent phenomena.**
Abstraction	**2.3.2**	**Use models and simulations to formulate, refine and test hypothesis.**
Data	**3.1.1**	**Use computers to process information, find patterns and test hypothesis about digitally processed information to gain insight and knowledge.**
Data	**3.1.2**	**Collaborate when processing information to gain insight and knowledge.**
Data	**3.1.3**	**Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.**
Data	**3.2.1**	**Extract information from data to discover and explain connections, patterns or trends.**
Data	**3.2.2**	**Use large data sets to explore and discover information and knowledge.**
Data	**3.3.1**	**Analyze how data representation, storage, security, and transmission of data**

		involve computational manipulation of information.
Algorithms	4.2.1	Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time.
Programming	5.1.2	Develop a correct program to solve problems.
Internet	6.3.1	Identify existing cybersecurity concerns and potential options that address these issues with the Internet and the systems built on it.
Impact	7.4.1	Explain the connections between computing and economic, social and cultural contexts.

Sample Assignment: 4

Part 1: View the following videos.

Bill Nye and a scaled model of the solar system (4:17)
https://www.youtube.com/watch?v=97Ob0xR0Ut8

Computer Generated Model of a Solar System (2:41)
https://www.youtube.com/watch?v=8z5mwAlxBYc

Part 2: Each student creates a journal entry responding to these two questions:

What was the main idea presented by each video?

What aspect(s) of the models helped make that point?

Part 3: Students discuss each of the following with elbow partners then groups.

How do the models in these videos depend on computing?

Consider the strengths and weaknesses of each model. What understanding can be better drawn from the first model and what understanding can better be drawn from the second?

What questions could be answered using these two models?

Part 4: From each group students share at least one response to each prompt.

Associated Computational Practices: P1, P3 and P5

Associated Big Ideas: BI2, BI3, BI7

Learning Objectives and Associated Computational Practices:

LO 2.3.1 [P3]

LO 2.3.2 [P3]

LO 3.1.3 [P5]

LO 7.4.1 [P1]

5. Data Manipulation

The main idea of Unit 5 is to create a synthesis of algorithms, programming, and data. Python lists will be used to work with a large set of data using searching, sorting, and other ways to manipulate data. Models and simulations will be used to represent real-life situations. Algorithms will be compared for their effectiveness as well as their readability.

Unit 5 Lessons:

5 - 1. Manipulating Large Data Sets
5 - 2. Searching
5 - 3. Sorting
5 - 4. Comparing Algorithms
5 - 5. Advanced Algorithms
5 - 6. Create Performance Task Partial Practice

Unit 5 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.3**	**Create a new computational artifact by combining or modifying existing artifacts.**
Creativity	**1.2.4**	**Collaborate in the creation of computational artifacts.**
Creativity	**1.2.5**	**Analyze the correctness, usability, functionality, and suitability of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.1.1**	**Describe the variety of abstractions used to represent data.**
Abstraction	**2.2.1**	**Develop an abstraction when writing a program or creating other computational artifacts.**
Abstraction	**2.2.2**	**Use multiple levels of abstraction to write programs.**
Abstraction	**2.2.3**	**Identify multiple levels of abstractions used when writing programs.**
Abstraction	**2.3.1**	**Use models and simulations to represent phenomena.**
Abstraction	**2.3.2**	**Use models and simulations to formulate, refine and test hypothesis.**
Data	**3.1.2**	**Collaborate when processing information to gain insight and knowledge.**
Data	**3.1.3**	**Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.**
Data	**3.2.1**	**Extract information from data to discover and explain connections, patterns or trends.**
Data	**3.2.2**	**Use large data sets to explore and discover information and knowledge.**
Algorithms	**4.1.1**	**Develop an algorithm for implementation in a program.**
Algorithms	**4.1.2**	**Express an algorithm in a language.**

Algorithms	4.2.1	Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time.
Algorithms	4.2.2	Explain the difference between solvable and unsolvable problems in computer science.
Algorithms	4.2.3	Explain the existence of undecidable problems in computer science.
Algorithms	4.2.4	Evaluate algorithms analytically and empirically for efficiency, correctness and clarity.
Programming	5.1.1	Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
Programming	5.1.2	Develop a correct program to solve problems.
Programming	5.1.3	Collaborate to develop a program.
Programming	5.2.1	Explain how programs implement algorithms.
Programming	5.3.1	Use abstraction to manage complexity in programs.
Programming	5.4.1	Evaluate the correctness of a program.
Programming	5.5.1	Employ appropriate mathematical and logical concepts in programming.
Internet	6.1.1	Explain the abstractions in the Internet and how it functions.
Impact	7.1.2	Explain how people participate in a problem solving process that scales.

Computational Thinking Practices used in Unit 5: P1, P2, P3, P4, P5 and P6

Sample Assignment 5

Read The Rise of Big Data in chunks: An Introduction to “Big Data” (20 mins) Reading can be found at: http://www.foreignaffairs.com/articles/139104/kenneth-neil-cukier-and-viktor-mayer-schoenberger/the-rise-of-big-data

Break students into groups or pairs and jigsaw the seven units of the reading. Each group is to summarize their section in a tweet sized comment (not more than 140 characters). Share tweets with the class.

Explain to students that big data is impacting every area of life. By using more data and processing power we can make better decisions. As an illustration, show a clip from the movie Moneyball: (3 mins) https://www.youtube.com/watch?v=rMObWsKaIls

After students watch, they create a journal entries explaining at least two ways data was used to better manage the baseball team. Partners discuss journal entries.. Share at least one observation with table groups and then share at least one observation from each group with the class.

Associated Computational Practices: P1, P4 and P5

Associated Big Ideas: BI1, BI3

Learning Objectives and Associated Computational Practices:

LO 1.2.5 [P4]

LO 3.1.1 [P4]

LO 3.1.3 [P5]

LO 3.2.1 [P1]

6. Data Visualization

The main idea of Unit 6 is to explore ways to create and understand data through data visualization. Students will also prepare for the final performance task by doing a group project.

Unit 6 Lessons:

At least one of these three lessons will be completed. Each lesson is scheduled for multiple days.

6 - 1. EarSketch
6 - 2. Data Visualization with Python and Bokeh
6 - 3. Dataquest

Unit 6 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.3**	**Create a new computational artifact by combining or modifying existing artifacts.**
Creativity	**1.2.4**	**Collaborate in the creation of computational artifacts.**
Creativity	**1.2.5**	**Analyze the correctness, usability, functionality, and suitability of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.2.1**	**Develop an abstraction when writing a program or creating other computational artifacts.**
Abstraction	**2.2.2**	**Use multiple levels of abstraction to write programs.**
Abstraction	**2.2.3**	**Identify multiple levels of abstractions used when writing programs.**
Data	**3.1.2**	**Collaborate when processing information to gain insight and knowledge.**
Data	**3.1.3**	**Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.**
Data	**3.2.1**	**Extract information from data to discover and explain connections, patterns or trends.**
Algorithms	**4.1.1**	**Develop an algorithm for implementation in a program.**
Algorithms	**4.1.2**	**Express an algorithm in a language.**
Algorithms	**4.2.4**	**Evaluate algorithms analytically and empirically for efficiency, correctness and clarity.**
Programming	**5.1.1**	**Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.**
Programming	**5.1.2**	**Develop a correct program to solve problems.**
Programming	**5.1.3**	**Collaborate to develop a program.**
Programming	**5.2.1**	**Explain how programs implement algorithms.**
Programming	**5.3.1**	**Use abstraction to manage complexity in programs.**

Programming	**5.4.1**	**Evaluate the correctness of a program.**
Programming	**5.5.1**	**Employ appropriate mathematical and logical concepts in programming.**
Internet	**6.1.1**	**Explain the abstractions in the Internet and how it functions.**

Computational Thinking Practices used in Unit 6: P1, P2, P3, P4, P5 and P6

Sample Assignment 6

Create a new EarSketch project. With your partner, develop an algorithm to meet the specifications below and to create music you like.

Length: 20 measures or longer Tempo: any
Structure

Your project should have an Intro, A and B sections, and an outro.
Use at least 3 audio clips in each section.
Have a repeated pattern such as ABAB.
Have an “Intro” section that is a unique beginning.
Have an “Outro” section that is unique to end the composition.
Use multiple effects including 4-parameter and the 7-parameter setEffect()

Individually develop separate programs to implement the algorithm.
Required Documentation:

Include your name, and a project description in the comments at the top of the file. Declare and use your own variables that uses audio clip constants as data abstractions for clips from the EarSketch Audio Loop browser.

Partners compare the two implementations and identify at least one important difference. Each individual should explain why they implemented the program as they did and compare their approach with their partner's.

Associated Computational Practices: P2, P3, P5 and P6

Associated Big Ideas: BI1, BI2, BI4, BI5

Learning Objectives and Associated Computational Practices:

LO 1.2.3 [P2]

LO 1.2.4 [P6]

LO 2.2.1 [P2]

LO 2.2.2 [P3]

LO 4.1.1 [P2]

LO 4.1.2 [P5]

LO 5.1.1 [P2]

LO 5.1.3 [P6]

Performance Task: **Create - Applications from Ideas.**

Students will have 12 hours (720 minutes) of class time to complete this performance task.

Advanced Placement
Computer Science Principles
Syllabus

December 2016

Instructor

Teacher:
School:

Course Overview

CS Matters in Maryland (http://csmatters.org/) is an NSF-funded CS10K project focusing on the new AP Computer Science Principles (CSP) course. The objectives of the course are consistent with those defined within the AP Computer Science Principles Curriculum Framework by the College Board. This course will be taught using the curriculum developed by CS Matters.

The CS Matters CSP curriculum was developed with one key goal: to provide all students the opportunity to learn computer science within a rigorous and engaging framework. To reach, retain, and teach traditionally underrepresented groups, the curriculum is designed to foster welcoming learning environments that are respectful of the diverse strengths of all students. The theme of the CS Matters course is *data* -- where it comes from, how it is collected and made available, how it can be analyzed and visualized, and the impact of "big data" on society.

Unit 1, *Your Virtual World*, informs and involves students in the many ways in which computing shapes their environment. Students study scalable problem solving by participating in citizen science, contributing their thoughts and recording their reactions in daily journals, investigating innovations of particular importance to them, and collaborating with partners and groups. Core lessons from Unit 1 include Impact of Innovation, A Problem Solving Process that Scales, Societal Impact, and Privacy in the Age of Big Data.

Unit 2, *Developing Programming Tools*, introduces students to software development using the Python programming language. The unit begins by focusing on the motivation for programming and then teaches the fundamentals of procedural programming, including data storage and retrieval, sequence, selection, iteration, and functions. This unit plays a pivotal role that allows subsequent units to challenge students to implement their own code to investigate their virtual world.

Unit 3, *Information and the Internet*, continues the emphasis on impact while examining the Internet, its core technologies, and its design. The unit explores how the design and technologies of the Internet affect innovation. Students take on the roles of Internet technologies by acting out the parts these technologies play. The first and third units together equip students to complete the College Board's *Explore* Performance Task.

In the Explore Performance Task (EPT), students choose and explore a computing innovation. The EPT requires students to select and investigate a computational innovation that: has or has had the potential to have significant beneficial and harmful effects on our society, economy, or culture, consumes, produces, and/or transforms data and raises at least one data storage concern, data privacy concern, or data

security concern. Students will have 8 hours (480 minutes) of class time to complete the EPT.

Unit 4, *Data Acquisition*, focuses on data, modeling, and simulations, while introducing fundamental concepts of probability and statistics. Students address the potential and limits of modeling by developing and testing hypotheses. Using computational thinking and the programming skills they learned in the prior unit, students build and test a model that leverages the power of computing to increase the accuracy of its results.

Unit 5, *Data Manipulation*, orients students to the conceptual foundations and core strategies for managing big data. Students investigate several data manipulation strategies, focusing on common algorithms and methods of evaluating them. The study of algorithms leads to small individual programming projects that acquaint students with the College Board's *Create* performance task.

Unit 6, *Data Visualization*, serves as a bridge between the introduction to computing and the development of more substantial programming artifacts. This unit includes several options for teachers to strengthen their students' creative programming abilities. The first lessons use EarSketch to engage students in computation through collaborative music composition. Other lessons use Bokeh from Continuum Analytics to equip students to create their own data visualizations.

In the Create Performance Task (CPT), students bring ideas to life through software development. The CPT requires students to design and iteratively develop a program. The College Board encourages but does not require students to collaborate with a partner, and once students begin the development of the CPT, they may receive help only from the collaborative partner. While not requiring it, this lesson supports a collaborative effort. Even if collaboration is used, each student must independently complete a significant level of planning, designing, and developing the program. Students will have 12 hours (720 minutes) of class time to complete the CPT.

Instruction in the Six Computational Practices and Seven Big Ideas is integrated throughout the course. The following table indicates the number of times each learning objective is addressed in each unit. Each Big Idea is taught in at least three units. Most Big Ideas are taught in every unit. The table below indicates the number of times lesson objectives associated with each big idea are taught in each unit. These are in addition to the instructional values of the Performance Tasks.

Summary of Big Ideas Taught in Each Unit							
	Big Ideas						
Unit	1	2	3	4	5	6	7
1	2	2	1	3	0	0	9
2	11	1	1	3	31	0	2
3	11	11	3	7	1	20	8
4	6	9	20	2	9	0	0
5	3	6	7	9	0	1	0
6	27	20	4	6	33	4	4

Six Computational Thinking Practices are used by students in each unit of the course.

- P1: Connecting Computing
- P2: Creating Computational Artifacts
- P3: Abstracting
- P4: Analyzing Problems and Artifacts
- P5: Communicating
- P6: Collaborating

Seven Big Ideas guide student activities throughout the course.

- BI1: Creativity
- BI2: Abstraction
- BI3: Data and Information
- BI4: Algorithms
- BI5: Programming
- BI6: The Internet
- BI7: Global Impact

Sample Assignment Associations			
Sample Assignments	Page Numbers	Computational Practices	Big Ideas
1	5	1, 4 and 6	1, 6 and 7
2	7	2, 4 and 6	1, 4 and 5
3	11	1, 2, 3, 4 and 5	2 and 6
4	14	1, 3 and 5	2, 3 and 7
5	17	1, 4 and 5	1 and 3
6	20	2, 3, 5 and 6	1, 2, 4 and 5

Course Content by Unit

1. **Your Virtual World**

The main idea of Unit 1 is to explore the effect of data on students' lives. Students will discover what is known about them online, explore the issue of data privacy, and the rights to access or change that data. After finding ways that the world of information is changing, students will learn how data is stored using binary codes, hardware and files.

Unit 1 Lessons:

- 1 - 1. Into the Darkness: A World Without Digital Communication
- 1 - 2. Into the Light: How Computers and the Internet Enhance Innovation
- 1 - 3. Exploring Innovations
- 1 - 4. It's Just Bits
- 1 - 5. How Innovation Affects Our Lives
- 1 - 6. A Problem Solving Process that Scales
- 1 - 7. Unit 1 Assessment
- 1 - 8. Practice Performance Task

Unit 1 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.3**	**Create a new computational artifact by combining or modifying existing artifacts.**
Creativity	**1.2.4**	**Collaborate in the creation of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.1.1**	**Describe the variety of abstractions used to represent data.**
Abstraction	**2.1.2**	**Explain how binary sequences are used to represent digital data.**
Data	**3.1.1**	**Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.**
Data	**3.2.1**	**Extract information from data to discover and explain connections or trends.**
Data	**3.2.2**	**Determine how large data sets impact the use of computational processes to discover information and knowledge.**
Algorithms	**4.1.1**	Develop an algorithm for implementation in a program.
Internet	**6.1.1**	**Explain the abstractions in the Internet and how it functions.**
Internet	**6.2.2**	**Explain how the characteristics of the Internet influence the systems built on it.**
Impact	**7.1.1**	**Explain how computing innovations affect communication, interaction and cognition.**
Impact	**7.1.2**	**Explain how people participate in a problem solving process that scales.**
Impact	**7.2.1**	**Explain how computing has impacted innovations in other fields.**
Impact	**7.3.1**	**Analyze the beneficial and harmful effects of computing.**
Impact	**7.4.1**	**Explain the connections between computing and economic, social and cultural contexts.**
Impact	**7.5.1**	**Evaluate online and print sources for appropriateness and credibility.**
Impact	**7.5.2**	**Access, manage, and attribute information using effective strategies.**

Computational Thinking Practices used in Unit 1: P1, P2, P4, P5 & P6

Sample Assignment 1

Students work in pairs to examine Wikipedia article, "Ten Commandments of Computer Ethics" (http://en.wikipedia.org/wiki/Ten_Commandments_of_Computer_Ethics) and identify and share the two commandments they think are the most commonly violated.
Pairs discuss how the commandments can be violated using cloud technology or social media.
Pairs prepare to share with the class how two rules are commonly violated. Identify the commandments commonly violated and the impact or consequence of each violation.
As a class, produce a revised version of the commandments.
Associated Computational Practices: P1, P4 and P6
Associated Big Ideas: BI1, BI6, BI7
Learning Objectives and Associated Computational Practices:

LO 1.2.4 [P6]
LO 6.3.1 [P1]
LO 7.1.1, [P4]
LO 7.3.1 [P4]
LO 7.4.1 [P1]

2. Developing Programs

The main idea of Unit 2 is to create solutions with code, debug, and verify results. Programming uses creative expression to solve problems correctly using abstraction, both individually and collaboratively. Students implement algorithms using math and logic, and evaluate programs for correctness.

The programming unit takes the formality of algorithm expression one step further by having students write programs in Python. In the programming unit, the levels of abstraction increase as students learn how a programming language can be used to control the machine on which the program is running. The topics of the programming unit are input/output, calculations with numbers, branching statements and Boolean logic, iteration, procedural abstraction, and processing data in a list.

We plan to use Python 3.x with the PyCharm IDE and the textbook *How to Think Like a Computer Scientist,* hosted by Runestone Interactive: (http://interactivepython.org/runestone/static/CCPS_Python/index.html)

Unit 2 Lessons:

2 - 1. Programming: Introduction and Motivation
2 - 2. Using Python and PyCharm
2 - 3. Algorithms: Basics
2 - 4. Algorithms: Pseudocode
2 - 5. Reading Python Code and Debugging
2 - 6. Types and Evaluation
2 - 7. Creating and Assigning Variables
2 - 8. Comparison, Logical Operators, and Conditional Execution
2 - 9. Nested and Chained Conditional Statements
2 - 10. Iteration: For Loops
2 - 11. Iteration: While Loops
2 - 12. Functions: Parameters and Return Values
2 - 13. Algorithms: Layers of Abstraction
2 - 14. Functions: Scope and Abstraction
2 - 15. Strings: Traversing, Slicing, and Parsing
2 - 16. Lists: Creation, Traversal, Insertion, and Removal
2 - 17. Unit Assessment

Unit 2 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.3**	**Create a new computational artifact by combining or modifying existing artifacts.**
Creativity	**1.2.4**	**Collaborate in the creation of computational artifacts.**
Creativity	**1.2.5**	**Analyze the correctness, usability, functionality, and suitability of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.1.1**	**Explain how binary sequences are used to represent digital data.**
Abstraction	**2.1.2**	**Identify multiple levels of abstractions used when writing programs.**
Data	**3.1.1**	**Use computers to process information, find patterns and test hypothesis about digitally processed information to gain insight and knowledge.**
Algorithms	**4.1.1**	**Develop an algorithm for implementation in a program.**
Algorithms	**4.1.2**	**Express an algorithm in a language.**
Programming	**5.1.2**	**Develop a correct program to solve problems.**
Programming	**5.1.3**	**Collaborate to develop a program.**
Programming	**5.2.1**	**Explain how programs implement algorithms.**
Programming	**5.3.1**	**Use abstraction to manage complexity in programs.**
Programming	**5.4.1**	**Evaluate the correctness of a program.**
Programming	**5.5.1**	**Employ appropriate mathematical and logical concepts in programming.**
Impact	**7.1.1**	**Explain how computing innovations affect communication, interaction and cognition.**
Impact	**7.4.1**	**Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts.**

Computational Thinking Practices used in Unit 2: P1, P2, P3, P4, P5 and P6

Sample Assignment 2

LinkyListy Role Play:

Create a human "list of students" by starting with an empty list the front of the room and then modifying the list by following the following program.

Designate a section of the board or a poster to act as the console/printer for output and a volunteer student to act as the printer driver.

Designate an area at the front of the room for the computer memory (and future students). The names should be changed to match those of students in the class. Also designate three students to play the role of fish and one as a dog.

As students are called by the program, have each student come up to stand in front of the room where memory resides. Have each student point to the student that follows them to form a list as they come to the front.

```
students = []
students.append('Joe')
students.append('Pat')
students.append('Alea')
students.append('Marta')
print(students)
#add additional commands to append or insert  4 or 5 more
students
print(students)
print(len(students))
print(students[3])
students.reverse()
print(students)
print('David' in students)
students.sort()
print(students)
more = ['Tom', 'Laverne']
students = students + more
print(students)
pets = ['fish'*3,'dog']
del students[1]
del students[2:4]
students.insert(0, 'Jennifer')
students = students + pets
print(students)
print(students.index('Marta'))
```

Associated Computational Practices: P2, P3, P4, P6

Associated Big Ideas: BI1, BI4, BI5

Learning Objectives and Associated Computational Practices:

LO 1.2.4 [P6]
LO 4.1.1 [P2]
LO 5.1. [P6]
LO 5.3.1 [P3]
LO 5.4.1 [P4]

3. Information and the Internet

The main idea of Unit 3 is to explore the Internet to prepare students to do research on a current Internet related topic in detail on their own. A variety of algorithms from different areas of interest are investigated. Students will learn how information is transmitted online and how search engines find and organize the data using complex algorithms. Studying cybersecurity and cryptography allows students to understand more about the issues concerning the privacy of data online which leads to a discussion on the ethics and social issues affecting the increased use of online data.

Unit 3 Lessons:

3 - 1. The Internet: Basics of Information Transmission
3 - 2. The Internet: Present and Future
3 - 3. How the Internet Works: Routing
3 - 4. How the Internet Works: Domain Name System
Optional 3 - 5. How the Internet Works: DNS Activity
3 - 6. Search Engines: Finding Information
3 - 7. Search Engines: Page Rank and Retrieval
Optional 3 - 8. Basic Statistics with Excel
3 - 9. Practice for Explore Performance Task
3 - 10. Cybersecurity: Attacks, Protection, and Impact
3 - 11. Cryptography: Symmetric Encryption
3 - 12. Cryptography: Public Key Encryption, Certificate Authorities, and Open Standards
Optional 3 - 13. Cybersecurity: Malicious Code, Identity Theft, and Remedies
3 - 14. Unit 3 Assessment

Unit 3 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.4**	**Collaborate in the creation of computational artifacts.**
Creativity	**1.2.5**	**Analyze the correctness, usability, functionality, and suitability of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.1.1**	**Describe the variety of abstractions used to represent data.**
Abstraction	**2.1.2**	**Explain how binary sequences are used to represent digital data.**
Data	**3.1.1**	**Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.**
Data	**3.1.3**	**Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language**
Data	**3.2.1**	**Extract information from data to discover and explain connections or trends**
Data	**3.3.1**	**Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information.**
Algorithms	**4.1.1**	**Develop an algorithm for implementation in a program.**
Algorithms	**4.2.2**	**Explain the difference between solvable and unsolvable problems in computer science.**
Programming	**5.1.1**	**Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.**
Programming	**5.1.2**	**Develop a correct program to solve problems.**
Programming	**5.3.1**	**Use abstraction to manage complexity in programs.**
Programming	**5.4.1**	**Evaluate the correctness of a program.**
Programming	**5.5.1**	**Employ appropriate mathematical and logical concepts in programming.**

Internet	**6.1.1**	**Explain the abstractions in the Internet and how it functions.**
Internet	**6.2.1**	**Explain characteristics of the Internet and the systems built on it.**
Internet	**6.2.2**	**Explain how the characteristics of the Internet influence the systems built on it.**
Internet	**6.3.1**	**Identify existing cybersecurity concerns and potential options that address these issues with the Internet and the systems built on it.**
Impact	**7.1.1**	**Explain how computing innovations affect communication, interaction and cognition.**
Impact	**7.3.1**	**Analyze the beneficial and harmful effects of computing.**
Impact	**7.4.1**	**Explain the connections between computing and economic, social and cultural contexts.**

Computational Thinking Practices used in Unit 3: P1, P2, P3, P4, P5 and P6

Sample Assignment 3

In this lesson, students will expand their knowledge of how the Domain Name System (DNS) works by acting as a class to simulate the use of DNS to retrieve web pages. Once the simulation is functioning students enhance its efficiency through the use of caching.
Poison the DNS cache by adding false DNS replies (DNS poisoning).
Students discuss with their groups how DNS works and how it supports Internet growth. Then they explain in their journals how:

- DNS works.
- Caching is both a benefit and a security risk.
- DNS supports Internet growth.

Associated Computational Practices: P1, P2, P3, P4, P5
Associated Big Ideas: BI2, BI6
Learning Objectives and Associated Computational Practices:

LO 2.3.1 [P3]
LO 4.1.1 [P2]
LO 6.1,1 [P3]
LO 6.2.1 [P5]
LO 6.2.2 [P4]
LO 6.3.1 [P1]

Performance Task: **Explore**. Research an innovation that uses the Internet.
Students will have 8 hours (480 minutes) of class time to complete this performance task.

4. Data Acquisition

The main idea of Unit 4 is to enable students to find data online and analyze it using a spreadsheet and Python code. To prepare for the performance task, students will follow the pattern of developing a hypothesis and then digging into data to find the answer. File input and output will allow students to use large sets of data, find patterns, look for metadata (data about data) and create visualizations of the data. Students will also investigate what is a reasonable solution and how the new data revolution can drive discovery and decision making.

Unit 4 Lessons:

4 - 1. Data Acquisition and Analysis
4 - 2. What are Models and Simulations?
4 - 3. Using Data and Simulations
4 - 4. File Input and Output using Python
4 - 5. Data Collection, Analysis, and Simulation
Optional 4 - 6. Hypothesis Testing with Simulations in NetLogo
4 - 7. Unit 4 Assessment

Unit 4 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.3**	**Create a new computational artifact by combining or modifying existing artifacts.**
Creativity	**1.2.4**	**Collaborate in the creation of computational artifacts.**
Creativity	**1.2.5**	**Analyze the correctness, usability, functionality, and suitability of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.1.1**	**Describe the variety of abstractions used to represent data.**
Abstraction	**2.2.3**	**Identify multiple levels of abstractions used when writing programs.**
Abstraction	**2.3.1**	**Use models and simulations to represent phenomena.**
Abstraction	**2.3.2**	**Use models and simulations to formulate, refine and test hypothesis.**
Data	**3.1.1**	**Use computers to process information, find patterns and test hypothesis about digitally processed information to gain insight and knowledge.**
Data	**3.1.2**	**Collaborate when processing information to gain insight and knowledge.**
Data	**3.1.3**	**Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.**
Data	**3.2.1**	**Extract information from data to discover and explain connections, patterns or trends.**
Data	**3.2.2**	**Use large data sets to explore and discover information and knowledge.**
Data	**3.3.1**	**Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information.**
Algorithms	**4.1.1**	**Develop an algorithm for implementation in a program.**
Algorithms	**4.1.2**	**Express an algorithm in a language.**
Programming	**5.1.2**	**Develop a correct program to solve problems.**
Internet	**6.3.1**	**Identify existing cybersecurity concerns and potential options that address these issues with the Internet and the systems built on it.**
Impact	**7.4.1**	**Explain the connections between computing and economic, social and cultural contexts.**

Sample Assignment: 4

Part 1: View the following videos.

Bill Nye and a scaled model of the solar system
(4:17) https://www.youtube.com/watch?v=97Ob0xR0Ut8
Computer Generated Model of a Solar System
(2:41) https://www.youtube.com/watch?v=8z5mwAlxBYc

Part 2: Each student creates a journal entry responding to these two questions:

What was the main idea presented by each video?
What aspect(s) of the models helped make that point?

Part 3: Students discuss each of the following with elbow partners then groups.

How do the models in these videos depend on computing?
Consider the strengths and weaknesses of each model. What understanding can be better drawn from the first model and what understanding can better be drawn from the second?
What questions could be answered using these two models?

Part 4: From each group students share at least one response to each prompt.

Associated Computational Practices: P1, P3 and P5
Associated Big Ideas: BI2, BI3, BI7
Learning Objectives and Associated Computational Practices:

LO 2.3.1 [P3]
LO 2.3.2 [P3]
LO 3.1.3 [P5]
LO 7.4.1 [P1]

5. Data Manipulation

The main idea of Unit 5 is to create a synthesis of algorithms, programming, and data. Python lists will be used to work with a large set of data using searching, sorting, and other ways to manipulate data. Models and simulations will be used to represent real-life situations. Algorithms will be compared for their effectiveness as well as their readability.

Unit 5 Lessons:

5 - 1. Manipulating Large Data Sets
5 - 2. Searching
5 - 3. Sorting
5 - 4. Comparing Algorithms
5 - 5. Advanced Algorithms
5 - 6. Create Performance Task Partial Practice

Unit 5 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.3**	**Create a new computational artifact by combining or modifying existing artifacts.**
Creativity	**1.2.4**	**Collaborate in the creation of computational artifacts.**
Creativity	**1.2.5**	**Analyze the correctness, usability, functionality, and suitability of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.1.1**	**Describe the variety of abstractions used to represent data.**
Abstraction	**2.1.2**	**Explain how binary sequences are used to represent digital data.**
Abstraction	**2.2.1**	**Develop an abstraction when writing a program or creating other computational artifacts.**
Abstraction	**2.2.2**	**Use multiple levels of abstraction to write programs.**
Abstraction	**2.2.3**	**Identify multiple levels of abstractions used when writing programs.**
Abstraction	**2.3.1**	**Use models and simulations to represent phenomena.**
Abstraction	**2.3.2**	**Use models and simulations to formulate, refine and test hypothesis.**
Data	**3.1.2**	**Collaborate when processing information to gain insight and knowledge.**
Data	**3.1.2**	**Collaborate when processing information to gain insight and knowledge.**
Data	**3.1.3**	**Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.**
Data	**3.2.1**	**Extract information from data to discover and explain connections, patterns or trends.**
Data	**3.2.2**	**Use large data sets to explore and discover information and knowledge.**
Algorithms	**4.1.1**	**Develop an algorithm for implementation in a program.**
Algorithms	**4.1.2**	**Express an algorithm in a language.**
Algorithms	**4.2.2**	**Explain the difference between solvable and unsolvable problems in computer science.**
Algorithms	**4.2.3**	**Explain the existence of undecidable problems in computer science.**
Algorithms	**4.2.4**	**Evaluate algorithms analytically and empirically for efficiency, correctness and clarity.**
Programming	**5.1.1**	**Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.**
Programming	**5.1.2**	**Develop a correct program to solve problems.**
Programming	**5.1.3**	**Collaborate to develop a program.**
Programming	**5.2.1**	**Explain how programs implement algorithms.**
Programming	**5.3.1**	**Use abstraction to manage complexity in programs.**
Programming	**5.4.1**	**Evaluate the correctness of a program.**

Programming	5.5.1	Employ appropriate mathematical and logical concepts in programming.
Internet	6.1.1	Explain the abstractions in the Internet and how it functions.
Impact	7.1.2	Explain how people participate in a problem solving process that scales.

Computational Thinking Practices used in Unit 5: P1, P2, P3, P4, P5 and P6

Sample Assignment 5

Read The Rise of Big Data in chunks: An Introduction to "Big Data" (20 mins) Reading can be found at: http://www.foreignaffairs.com/articles/139104/kenneth-neil-cukier-and-viktor-mayer-schoenberger/the-rise-of-big-data

Break students into groups or pairs and jigsaw the seven units of the reading. Each group is to summarize their section in a tweet sized comment (not more than 140 characters). Share tweets with the class.

Explain to students that big data is impacting every area of life. By using more data and processing power we can make better decisions. As an illustration, show a clip from the movie Moneyball: (3 mins) https://www.youtube.com/watch?v=rMObWsKaIls

After students watch, they create a journal entries explaining at least two ways data was used to better manage the baseball team. Partners discuss journal entries.. Share at least one observation with table groups and then share at least one observation from each group with the class.

Associated Computational Practices: P1, P4 and P5

Associated Big Ideas: BI1, BI3

Learning Objectives and Associated Computational Practices:

LO 1.2.5 [P4]
LO 3.1.1 [P4]
LO 3.1.3 [P5]
LO 3.2.1 [P1]

6. Data Visualization

The main idea of Unit 6 is to explore ways to create and understand data through data visualization. Students will also prepare for the final performance task by doing a group project.

Unit 6 Lessons:

At least one of these three lessons will be completed. Each lesson is scheduled for multiple days.

6 - 1. EarSketch
6 - 2. Data Visualization with Python and Bokeh
6 - 3. Dataquest
Optional: 6 - 4. Diversity Makes For Better Solutions

Unit 6 Learning Objectives:

Creativity	**1.1.1**	**Apply a creative development process when creating computational artifacts.**
Creativity	**1.2.1**	**Create a computational artifact for creative expression.**
Creativity	**1.2.2**	**Create a computational artifact using computing tools and techniques to solve a problem.**
Creativity	**1.2.3**	**Create a new computational artifact by combining or modifying existing artifacts.**
Creativity	**1.2.4**	**Collaborate in the creation of computational artifacts.**
Creativity	**1.2.5**	**Analyze the correctness, usability, functionality, and suitability of computational artifacts.**
Creativity	**1.3.1**	**Use computing tools and techniques for creative expression.**
Abstraction	**2.1.1**	**Describe the variety of abstractions used to represent data.**
Abstraction	**2.2.1**	**Develop an abstraction when writing a program or creating other computational artifacts.**
Abstraction	**2.2.2**	**Use multiple levels of abstraction to write programs.**
Abstraction	**2.2.3**	**Identify multiple levels of abstractions used when writing programs.**
Abstraction	**2.3.1**	**Use models and simulations to represent phenomena.**
Data	**3.1.1**	**Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.**
Data	**3.1.2**	**Collaborate when processing information to gain insight and knowledge.**
Data	**3.1.3**	**Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.**
Data	**3.2.1**	**Extract information from data to discover and explain connections, patterns or trends.**
Data	**3.2.2**	**Determine how large data sets impact the use of computational processes to discover information and knowledge.**
Algorithms	**4.1.1**	**Develop an algorithm for implementation in a program.**
Algorithms	**4.1.2**	**Express an algorithm in a language.**
Programming	**5.1.1**	**Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.**
Programming	**5.1.2**	**Develop a correct program to solve problems.**
Programming	**5.1.3**	**Collaborate to develop a program.**
Programming	**5.2.1**	**Explain how programs implement algorithms.**
Programming	**5.3.1**	**Use abstraction to manage complexity in programs.**
Programming	**5.4.1**	**Evaluate the correctness of a program.**
Programming	**5.5.1**	**Employ appropriate mathematical and logical concepts in programming.**
Impact	**7.1.1**	**Explain how computing innovations affect communication, interaction and cognition.**
Impact	**7.2.1**	**Explain how computing has impacted innovations in other fields.**
Impact	**7.3.1**	**Analyze the beneficial and harmful effects of computing.**
Impact	**7.4.1**	**Explain the connections between computing and real-world contexts,**

		including economic, social, and cultural contexts.

Computational Thinking Practices used in Unit 6: P1, P2, P3, P4, P5 and P6

Sample Assignment 6

Create a new EarSketch project. With your partner, develop an algorithm to meet the specifications below and to create music you like.
Length: 20 measures or longer Tempo: any
Structure

Your project should have an Intro, A and B sections, and an outro.
Use at least 3 audio clips in each section.
Have a repeated pattern such as ABAB.
Have an "Intro" section that is a unique beginning.
Have an "Outro" section that is unique to end the composition.
Use multiple effects including 4-parameter and the 7-parameter setEffect()

Individually develop separate programs to implement the algorithm.
Required Documentation:

Include your name, and a project description in the comments at the top of the file. Declare and use your own variables that uses audio clip constants as data abstractions for clips from the EarSketch Audio Loop browser.

Partners compare the two implementations and identify at least one important difference. Each individual should explain why they implemented the program as they did and compare their approach with their partner's.
Associated Computational Practices: P2, P3, P5 and P6
Associated Big Ideas: BI1, BI2, BI4, BI5
Learning Objectives and Associated Computational Practices:
LO 1.2.3 [P2]
LO 1.2.4 [P6]
LO 2.2.1 [P2]
LO 2.2.2 [P3]
LO 4.1.1 [P2]
LO 4.1.2 [P5]
LO 5.1.1 [P2]
LO 5.1.3 [P6]

Performance Task: **Create - Applications from Ideas.**
Students will have 12 hours (720 minutes) of class time to complete this performance task.

(http://csmatters.org) 0 - 1

0b0 - 0b1

How Does Technology Impact Your Life?

Unit 0. Introduction

Revision Date: Jun 30, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary: This optional lesson can be used as a brief preview to the course on the first day of class when teachers typically have many classroom administrative tasks to accomplish, but teachers also want to set the stage for the class. Students begin thinking about the influences of technology as they engage in a Think-Pair-Share strategy about the computing innovations that have impacted their lives. In addition, this is an opportunity to set beginning expectations about interacting with classmates in collaborative activities and to model writing in the content area.

Outcomes:

- Students will provide examples of how technology makes a difference in their lives.
- Students will describe both the positive and negative impacts of the computing innovation that has had the most impact on their life.

Overview:

1. Getting Started (5 min) - Journal Activity
2. Guided Activities (40 min) - Students Think-Pair-Share and engage in independent writing
3. Wrap Up (5 min) Summary and Homework Assignment

Source: This lesson is adapted from Code.org (Unit 1 Lesson 00)

Learning Objectives

CSP Objectives

- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
  - 7.1.1L - Computing contributes to many assistive technologies that enhance human capabilities.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
  - 7.3.1A - Innovations enabled by computing raise legal and ethical concerns.

This beginning lesson engages students in a discussion of how technology makes a difference in their lives and helps to develop the desired classroom dynamics and habits that will appear throughout the class.

Common Core ELA:

- WHST 12.1 - Write arguments on discipline specific content

Key Concepts

Innovation is an important theme of this course. Students will become more aware of how innovation in technology has affected their lives.

Essential Questions

- How does computing enhance human communication, interaction, and cognition?
- How does computing enable innovation?

Teacher Resources

Student computer usage for this lesson is: **none**

Student journals, if they are ready. Otherwise, paper for students to write on.

Lesson Plan

Getting Started (5 min)

This short activity can be used after any required first-day administrative duties.

Introducing Journals and "Think-Pair-Share" (5 min)

Each lesson in the course will start out with a brief prompt for students to respond to in their journals. They will pair off to discuss their answers, then share their findings with the class.

If you have the journals ready, have students write the answer to this prompt in the journal: Identify technological innovations that you and/or your families use. (At LEAST four, how many can you name?)

If Clarification is Needed:

- Invention: a completely new product or idea
- Innovation: an improvement to an existing idea
- Possibly use http://www.pbs.org/idealab/2012/03/the-difference-between-invention-and-innovation086/ (http://www.pbs.org/idealab/2012/03/the-difference-between-invention-and-innovation086/) (Read the 2nd section)

Guided Activities (40 min)

Activity - Think-Pair-Share (20 min)

1. Have students share journal ideas in pairs. Each student should circle the four most interesting ideas in their journal.
2. Have pairs combine into groups of 4-6 and write 4-6 unique, interesting ideas from their group on post-it notes or paper.
3. Display student ideas at the front of the room on a board or poster using pen, chalk or post-its.
   (Suggestion: Take turns letting each group contribute one thing at a time and explain how it's used in their lives. Either disallow duplicates or notice how many groups choose the same things.)
4. Challenge students as a group to think of at least 10 more things that are not on the list and add them.

Activity - Independent Activity (20 min)

1. Tell students to complete this sentence: "The computing innovation that has had the most impact on my life is .... because ....".
2. Write the completed sentence and describe both the positive and negative impacts of the innovation in a short paragraph in your journal.
3. Allow students a few minutes to independently and silently think and write.
4. If there is time, join each student pair with another pair for a second round of sharing and discussion.
5. *Suggestion: A large-group class discussion can replace the second pair-share portion of the activity if it is more appropriate for the class setting or the time available.*

Note: Written communication is an important skill. This curriculum provides a variety of opportunities for students to develop the skills that they will need to perform well in the Performance Tasks.

Wrap-Up (5 min)

Summarize the various ways that computing innovation has affected our lives. Assign the following homework.

Homework

Ask students to interview an adult and ask, "What computing innovation has had the most impact on your life? In what ways has your life been affected?"

Students will record the adult's answer and compare and contrast the answer with their own original answer in a brief paragraph.

Options for Differentiated Instruction

Optional Extension:

Suggest that students create timelines showing the years when the various innovations were invented or became available to consumers. (They could make their best guesses as a class and reorder the ideas on the board.)

Evidence of Learning

Summative Assessment

Paragraph about the positive and negative impacts of the innovation that has had the most impact on the student's life in journal at the end of class.

Paragraph for homework that compares and contrasts an adult's answer to their own journal entry answer after conducting a discussion with an adult at home.

(http://csmatters.org) 0 - 2

0b0 - 0b10

Computing for the Next Generation

Unit 0. Introduction

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

Through presentations, videos, and discussion, students discover how technology has been changing and brainstorm ideas for how the next generation will have a different relationship with technology than the current generation. Students will also learn the relative measures of computer storage (KB, MB, etc.)

This is a prelude to the idea of big data and the impact of technology.

Outcomes

- Students describe some of the ways that technology has been changing
- Students brainstorm ideas for how the next generation will have a different relationship with technology than the current generation
- Students compare the relative measures of computer storage

Overview

1. Introduction (5 min) - Students journal about the different relationships each generation has with technology.

2. Activities (40 min) - Students practice organizing by relative size and create posters to demonstrate learning.

3. Wrap-up (5 min) - Posters are shared with the class.

Learning Objectives

CSP Objectives

- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4E - Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.
  - 1.2.4F - A collaboratively created computational artifact can reflect personal expressions of ideas.
- 1.2.5 - Analyze the correctness, usability, functionality, and suitability of computational artifacts.
  - 1.2.5C - The functionality of a computational artifact may be related to how it is used or perceived.
  - 1.2.5D - The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
  - 7.1.1C - Social media continues to evolve and fosters new ways to communicate.
  - 7.1.1D - Cloud computing fosters new ways to communicate and collaborate.
  - 7.1.1E - Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results.
  - 7.1.1H - Social media, such as blogs and Twitter, have enhanced dissemination.
  - 7.1.1K - Smart grids, smart buildings, and smart transportation are changing and facilitating human capabilities.
  - 7.1.1L - Computing contributes to many assistive technologies that enhance human capabilities.
- 7.1.2 - Explain how people participate in a problem solving process that scales.
  - 7.1.2C - Human computation harnesses contributions from many humans to solve problems related to digital data and the Web.

    - 7.1.2F - Crowdsourcing offers new models for collaboration, such as connecting people with jobs and businesses with funding.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
    - 7.3.1J - Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.
    - 7.3.1K - People can have instant access to vast amounts of information online; accessing this information can enable the collection of both individual and aggregate data that can be used and collected.

Essential Questions

- How can computing and the use of computational tools foster creative expression?
- How can computing extend traditional forms of human expression and experience?
- How does computing enhance human communication, interaction, and cognition?
- How does computing enable innovation?

Teacher Resources

Student computer usage for this lesson is: **none**

Videos:

- Oracle The Information Age https://www.youtube.com/watch?v=WFgT9KCxQ0k (https://www.youtube.com/watch?v=WFgT9KCxQ0k) (3:51)
- https://www.youtube.com/watch?v=qGYmML0e0X4 (https://www.youtube.com/watch?v=qGYmML0e0X4) (2:31 - stop at 1:14)
- https://www.youtube.com/watch?v=6yWzKvQXsYM (https://www.youtube.com/watch?v=6yWzKvQXsYM) Technology In 2019 - What The Future Of Tech Looks Like! (5:51) Published on Mar 11, 2013

PowerPoints (in the Lesson Resources folder):

1. Unit0_Lesson2_The Information Age How Big.pptx
2. Unit0_lesson2_TheFuture.ppt

Students should have paper for taking notes. (If desired, use preprinted Information Age Notes, in the Lesson Resources folder)

Print out "Relative Measure Word Strips to cut" papers and "Measure Quantity Word Strips to cut" papers (in the Lesson Resources folder) and cut them into strips to give to students for the activity.

8 big sheets of big poster paper to place at the front of the room labeled:

1. communication
2. education
3. automation (robotics)
4. privacy
5. entertainment
6. transportation
7. medicine & health
8. the Internet

Post-it notes for students

Lesson Plan

Introduction - Journal and Discussion (5 min)

1. Journal: How were computers and technology different one generation ago?

2. Pair and share journal ideas, compile a class list. *(Ideas you might suggest if they're stuck: bigger, slower, less storage, less portable, not inside of so many other things, not as pervasive, less voice ability, less connection to the Internet, phones weren't very smart, more expensive, Previous generation: land lines, stand in one place when talking on the phone, separate cameras, internet via phone modem, no GPS, etc.)*

3. Prompt students to come up with their own definition of what "1 generation ago" means. (Now a generation is considered to be about 25 years. A century ago it was about 20 years - from Ancestry.com)
{optional question for thought/discussion: Why were generations shorter a hundred years ago? (don't tell them the answer, try to lead them to ask each other good questions to guide them to an answer: shorter lifespans, more death from disease, younger marriage age
http://www.ancestry.com.au/learn/learningcenters/default.aspx?section=lib_Generation
(http://www.ancestry.com.au/learn/learningcenters/default.aspx?section=lib_Generation))}

Activities: (40 min)

Part 1 (3-4 min)

Hand out mixed up **relative measure papers** (a document with strips to cut is in the Lesson Resources folder) to 8 students.

- In order: kilobyte, megabyte, gigabyte, terabyte, petabyte, exabyte, zettabyte, and yottabyte.

Ask them to try to line up by relative size at the front of the room to display the terms from smallest to largest. Ask if the class agrees, make changes by group vote. Don't tell them if they're right yet. Tape or post the strips to the front wall.

Teacher Note: Be sure to get students' names and introduce them as part of the goal of developing a classroom community of learners. Continuously encourage positive social interactions.

Part 2 (3-4 min)

Hand out the 8 papers with **measured quantities** (a document with strips to cut is in the Lesson Resources folder).

- In order: a picture the size of your fingernail, a small novel, a symphony recorded in high fidelity sound, the whole library of congress, 5 years' worth of the data recorded by NASA earth orbit satellite, all the words ever spoken by humankind written down, all recorded TV broadcasts and movies stored as video, and amount of data the National Security Administration can store.

Ask those students to try to match themselves up with the relative measure papers. Give the class a chance to rearrange by group vote. Tape or post the strips to their relative measure paper.

Part 3 (10 min)

1. Present the PowerPoint: **Unit0_Lesson2 The Information Age. How Big.**
    - Have students take notes on the correct sequence of relative measures. (Use blank paper or Student notes for The Information Age How Big) Rearrange paper strips at the front of the room as needed to match the true order. (5 min)
2. Show the video: **The information age. Big Data is Changing the World** (3:51)
    - https://www.youtube.com/watch?v=WFgT9KCxQ0k (https://www.youtube.com/watch?v=WFgT9KCxQ0k)
3. Have groups try to guess how many people used the Internet each day in 2000 and now, how many searches were done each day on Google in 1998 and now. (3 min)
4. Show the video: 2:31 https://www.youtube.com/watch?v=qGYmML0e0X4 (https://www.youtube.com/watch?v=qGYmML0e0X4) Exalogic: Ready for the Future by Oracle (only show the first 1:14)

Part 4 (20 min)

1. Present the PowerPoint **Unit0_Lesson2 The Future.**
2. Group students in 3's or 4's. (max of 8 groups)
    - Groups **brainstorm**: What new technology might the next generation have?
        - Ask them to think of how technology will be different for the next generation in each of these categories: communication, education, automation (robotics), transportation, medicine and health, the Internet, privacy, and entertainment. Write at least 10 ideas for each group on post it notes, 1 idea per post it. (5 min)
        - **Note:** 1 generation is about 25 years, so the previous generation was born around 1975 and went to high school around 1990. You were born around 2000, and are in high school now. The next generation will be born around 2025, and be in high school around 2040.
3. Place large **posters** at the front of the room. Take turns having groups contribute 1 post it note at a time and explain their ideas. (5 min)
    - Groups get 1 point for each unique idea that gets posted.
    - The group with the most points, when all notes have been posted, chooses the poster they want to work on. The group with the 2nd highest score chooses next, etc.
4. Groups design a visual artifact on their poster: a labeled drawing, concept map, cartoon (stick figures are fine), or some other visual to describe life for the next generation. Encourage students to do something visually appealing, creative, interesting, or informative. (10 min)

Wrap-Up (5 min)

Display the posters created by the class, share details and ideas from the posters.

Additional Activities if Time Permits:

1. Show a video on a possible work world of the future: https://www.youtube.com/watch?v=t5X2PxtvMsU (https://www.youtube.com/watch?v=t5X2PxtvMsU) (5:51)
2. Discuss what ideas are the same/different from the ideas that students shared.
3. Have a vote on the posters for most creative, most unique, best detail, etc.

Options for Differentiated Instruction

Instead of having students move to the front of the room to sort the paper strips into order, you could print a set for each group of 3-4 students and have them match and arrange them at their desks.

If students have a strong interest, or if you have extra time, use the ideas in "Extensions to the lesson on the Future of Technology" document located in the resources folder.

Evidence of Learning

Formative Assessment

Self-checking exercise on identifying storage terms (KB, MB, etc.)

(http://csmatters.org) 0 - 3

0b0 - 0b11

Intelligent Paper

Unit 0. Introduction

Revision Date: Jun 29, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary: This lesson is a basic introduction to algorithms and the nature of intelligence. Students will play tic-tac-toe (noughts and crosses is the British version) between a “highly intelligent piece of paper” and a human. Students will explore how to create an algorithm and the concept of computer intelligence.

Outcomes

- Students will play tic-tac-toe using a specific algorithm.
- Students will create and test a new algorithm for playing tic-tac-toe.

Overview

1. Getting Started (5 min) - Journal
2. Introduction (5 min) - PowerPoint
3. Guided Activities (35 min) - Students play tic-tac-toe with a given algorithm, then practice designing and implementing their own.
4. Wrap-up (5 min) - Students create definitions and assign homework for Lesson 1-1
5. Optional Activity (5 min) - Students examine an actual "AI" (simple game playing program) created for tic-tac-toe in Python

Source: This lesson is adapted from a lesson created by Paul Curzon, Queen Mary, University of London.

Learning Objectives

CSP Objectives

- 3.2.1 - Extract information from data to discover and explain connections or trends
  - 3.2.1B - Large data sets provide opportunities for identifying trends, making connections in data, and solving problems.
  - 3.2.1C - Computing tools facilitate the discovery of connections in information within large data sets.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
  - 7.1.1M - The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.
  - 7.1.1N - The Internet and the Web have changed many areas, including e-commerce, health care, access to information and entertainment, and online learning.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
  - 7.3.1A - Innovations enabled by computing raise legal and ethical concerns.
  - 7.3.1H - Aggregation of information, such as geolocation, cookies, and browsing history, raises privacy and security concerns.
  - 7.3.1J - Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.

- 7.3.1K - People can have instant access to vast amounts of information online; accessing this information can enable the collection of both individual and aggregate data that can be used and collected.

Common Core ELA:

- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes

NGSS Practices:

- 8. Obtaining, evaluation, and communicating information

Essential Questions

- How do computer programs implement algorithms?
- How does computing enhance human communication, interaction, and cognition?

Teacher Resources

Student computer usage for this lesson is: **none**

A PowerPoint for this lesson is included in the Lesson Resources folder - IntelligentPaper.pptx and IntelligentPaper.pdf

Copies for student pairs of "intelligent paper directions" with tic-tac-toe directions on one side, and blank on the other - in the Lesson Resources folder - IntelligentPaperDirections.pdf

The wrap-up questions are available in the Lesson Resources folder as Questions To Consider.docx

Optional: a musical greeting card, a paper folded into a fortune teller (http://en.wikipedia.org/wiki/Paper_fortune_teller (http://en.wikipedia.org/wiki/Paper_fortune_teller)), a page of equations

The Python program for the optional activity is located in the Lesson Resources Folder - TicTacToeAI.py

Lesson Plan

Getting Started (5 min) - Journal

What could make a piece of paper intelligent? (Think-Pair-Share)

Introduction of Content (5 min)

(Use IntelligentPaper.pptx in the Lesson Resources folder to help deliver this lesson.)

Challenge the students by saying that you have a piece of paper that is at least as smart as any human. (Show the blank side of the paper, don't tell the students yet, but it has directions on how to play tic-tac-toe on the back.) Ask if anybody believes that this is possible.

Show students examples of "smart papers," such as:

- A page full of equations
- A paper folded into a fortune teller http://en.wikipedia.org/wiki/Paper_fortune_teller (http://en.wikipedia.org/wiki/Paper_fortune_teller)
- A greeting card that plays music when you open it. It's made of paper, is it intelligent? (it's the chip inside that's smart, and somebody had to program it to make it do something... same with any paper, a person needs to put the information on there to "give" it intelligence)

Encourage discussion and debate, prod students to argue their point for or against intelligence, and get them to develop their own criteria and definition for intelligence. Write the class definition and criteria on the board.

Guided Activities (35 min)

Part 1 - Game Activity (5 min)

Tell the class that the paper has never lost a game: it has perfect intelligence.

Challenge students to play a game against the paper. The paper is peripherally challenged (it has no arms, and thus needs somebody to do its work for it). One person represents humankind, while the other person represents the paper. Play tic-tac-toe with a partner. The paper must begin the game.

Possible outcomes:

- Paper wins or draws.
- Humans give up. (They often do.)
- Try again and see if it's just luck?

- Humans cheat. (They sometimes do.)
- Humans cause an error. (It happens.)

But, the paper WILL NOT LOSE.

Try letting humankind go first. (Wait and try it: The paper will lose. Why?)

Part 2 - Independent Activity (15 min)

Challenge students to write out detailed directions (an algorithm) that will never lose the game whether it goes first or second.

Part 3 - Game Activity with Student Algorithms (15 min)

Students should use their new algorithm to play against each other. Follow the same model for the paper versus the human game.

Discuss how testing is essential in order to figure out whether the algorithm works for every possible game.

Additional Possible Activities and Discussions with Time Permitting:

- There are lots of websites that show all of the possible moves in tic-tac-toe.
- If there is time, have students try to figure out how many different, unique games can possibly be played in tic-tac-toe.
- Point out, after they give it a good try, that using symmetry cuts out a lot of redundant possible moves.
- This can also lead to an investigation of how machine learning happens in artificial intelligence by looking at all possible moves and making decisions.
- How many possible tic-tac-toe games are there? How can you represent all the possibilities?
- Have students share how they diagrammed or kept track of how many different possible games there are
- You can show them some ideas if you just do an image search for tic-tac-toe:
    - For example, http://upload.wikimedia.org/wikipedia/commons/thumb/d/da/Tic-tac-toe-game-tree.svg/300px-Tic-tac-toe-game-tree.svg.png (http://upload.wikimedia.org/wikipedia/commons/thumb/d/da/Tic-tac-toe-game-tree.svg/300px-Tic-tac-toe-game-tree.svg.png)
- What about 3D tic-tac-toe? http://en.wikipedia.org/wiki/3-D_Tic-Tac-Toe (http://en.wikipedia.org/wiki/3-D_Tic-Tac-Toe)

Wrap-up (5 min)

Have students write their own definitions for the four words at the end of the presentation:

- Computer program
- Artificial intelligence
- Peripheral
- Algorithm

Optional Activity (5 min)

(Use PyCharm or some other Python environment to show the TicTacToeAI.py program from the Lesson Resources folder.)

- Follow slides 14-16 of the PowerPoint to review the Python "AI" written for TicTacToe. Encourage discussion on how a computer can "think" in order to win the game.
- Play a few rounds against the program and see whether a student can figure out how to beat it.
- Discuss how the program could have been better designed.

Homework

Assign homework for Lesson 1-1: Provide students a copy of the “Questions to Consider” in the resources folder and assign the reading:

Blown to Bits – Chapter 1, can be found here http://www.bitsbook.com/wp-content/uploads/2008/12/chapter1.pdf (http://www.bitsbook.com/wp-content/uploads/2008/12/chapter1.pdf) and is available in the lesson resources folder for Unit 0 Lesson 3.

Options for Differentiated Instruction

Extension: If you have extra time, have a championship contest between one set of student-generated instructions and another, alternating who goes 1st and 2nd. You can work in groups of three, with one person acting as the judge if desired.

Evidence of Learning

Formative Assessment

Vocabulary entries in journals from the end of the PowerPoint presentation

Group participation in interactive activity

Writeup about a more general solution

(http://csmatters.org) 1 - 1

0b1 - 0b1

Into the Darkness: A World Without Digital Communication

Unit 1. Your Virtual World

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 3 50-minute sessions

Lesson Summary

Pre-lesson preparation

Students must complete a pre-reading assignment (the first chapter of Blown To Bits, which is available online (http://www.bitsbook.com/wp-content/uploads/2008/12/chapter1.pdf) or in the Resources folder). This pre-reading can be assigned at the end of Unit 0.

Summary

Students will read about the "Digital Explosion" and discuss exponential growth. They will discuss and share insights on what a world without digital communication would be like and investigate some of the things that are possible because of digital communication. They will then share their findings with the class.

Outcomes

- Students will explain how innovation affects communication, interaction, and cognition.
- Students will explain how computing has impacted innovation in other fields.
- Students will analyze some of the effects of computing.
- Students will explain some connections between technology, and economic and social differences.

Overview

Session 1

1. Getting Started (5 min) - Pair activity on exponential growth
2. Activity (10 min) - Think-Pair-Share and independent writing.
3. Activity (35 min) - Teams brainstorm on their usage of digital communication and discuss how life would be different without it.
4. Homework - Each team member will write one reflection question, or interview 3 people about what would be different in a world without digital communication and write up what they learn.

Session 2

1. Getting Started (5 min) - Journal Activity
2. Activity (15 min) - Teams engage in algorithm / calculation activity without the benefit of digital tools.
3. Activity (30 min) Teams brainstorm and organize ideas about the impact of digital communication.
4. Homework - Students will write a reflection on how a particular aspect of society depends on computers.

Session 3

1. Getting Started / Activity (35 min) - Students will each write a short story about a world without digital communication.
2. Activity (15 min) - Students will journal about their social media post from Session 1, and share with a partner.

Learning Objectives

CSP Objectives

- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4E - Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.
  - 1.2.4F - A collaboratively created computational artifact can reflect personal expressions of ideas.
- 3.2.1 - Extract information from data to discover and explain connections or trends
  - 3.2.1B - Large data sets provide opportunities for identifying trends, making connections in data, and solving problems.
  - 3.2.1C - Computing tools facilitate the discovery of connections in information within large data sets.
- 6.1.1 - Explain the abstractions in the Internet and how the Internet functions.
  - 6.1.1A - The Internet connects devices and networks all over the world.
  - 6.1.1D - The Internet and the systems built on it facilitate collaboration.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
  - 7.1.1A - Email, SMS, and chat have fostered new ways to communicate and collaborate.
  - 7.1.1E - Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results.
  - 7.1.1H - Social media, such as blogs and Twitter, have enhanced dissemination.
- 7.2.1 - Explain how computing has impacted innovations in other fields.
  - 7.2.1F - Moore's law has encouraged industries that use computers to effectively plan future research and development based on anticipated increases in computing power.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
  - 7.3.1A - Innovations enabled by computing raise legal and ethical concerns.
- 7.4.1 - Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts.
  - 7.4.1A - The innovation and impact of social media and online access varies in different countries and in different socioeconomic groups.
  - 7.4.1B - Mobile, wireless, and networked computing have an impact on innovation throughout the world.
  - 7.4.1D - Groups and individuals are affected by the "digital divide" - differing access to computing and the Internet based on socioeconomic or geographic characteristics.

NGSS Practices:

- 5. Using mathematics and computational thinking
- 7. Engaging in argument from evidence
- 8. Obtaining, evaluation, and communicating information

NGSS Content:

- HS-ETS1-1. Analyze a major global challenge to specify qualitative and quantitative criteria and constraints for solutions that account for societal needs and wants.

Key Concepts

Chapter 1 of "Blown to Bits" and the lesson motivate students to begin thinking about the advancement of technology and its impact on many aspects of their lives (both positively and negatively). Subsequent lessons will research particular impacts on society in more depth.

Students will consider a world without digital communication to emphasize the impact that computers have on their everyday lives and how integral computers and digital communication have become to our ordinary existence.

Teacher Resources

Student computer usage for this lesson is: **none**

TEACHER RESOURCES

In the Lesson Resources Folder:

- *Blown to Bits* – Chapter 1 reading assignment
  - from: http://www.bitsbook.com/wp-content/uploads/2008/12/chapter1.pdf (http://www.bitsbook.com/wp-content/uploads/2008/12/chapter1.pdf)
- PowerPoints: What If Part 1, What If, Part 2, What If Part 3

Prior to the Lesson:

- Provide students a copy of the “Questions to Consider” (in the resources folder) and assign the *Blown to Bits* reading assignment (above)

During the lesson, students will need:

- Student Journal

Lesson Plan

Session 1

This lesson assumes that students have either taken a previous CS course or that you have done Unit 0, so that students know what a computer is, how to write a basic algorithm, and the basic history of technology. It also assumes that students have read Blown to Bits, Chapter 1.

Getting Started (5 min)

Have students work in pairs to discuss and answer the following questions. (If possible, provide students with calculators. An exact value is not required to formulate an answer. The choices reflect three different types of growth.)

Someone offers you a summer job with a choice of three pay rates:

1. $10 per hour for eight hours of work for day for 30 days.

2. One dollar the first day, two dollars the second day, three dollars the third day, and so on (increasing by one dollar each day).

3. One cent on day one, two cents on day two, four cents on day three, and so on (doubling each day for 30 days).

Which pay rate would you choose? Why? What does this illustrate?

Solution: After 30 days,

- The first choice nets 10*8*30 = $2,400.
- The second offer will pay $465.
- The third offer will pay 2 to the 30th power (minus 1) cents, which is over $10 million.

Clearly, the last choice is the best, even though it starts with the lowest value (although you are unlikely to receive such an offer!)

This activity illustrates *exponential growth* (which was discussed in the chapter in the context of data growth).

Activity (10 min) - Think-Pair-Share

Have students discuss with their partner the answers to the pre-reading questions from Blown To Bits Chapter 1 (see Questions to Consider in Teacher Resources).

Choose one of the more open-ended questions from the pre-reading questions. Either:

- give examples of things today that are stored in bits
- describe examples of innovations that are neither good nor bad
- list ways that life is more complicated because of the explosion of bits

Have partners pick their most interesting answers and post them or write them on something in the front of the room to share with the class.

Activity (35 min) – Form Teams and Investigate Communication and Digital Communication

Use a creative method for dividing students up into teams of 3-4 (line up by birthday, etc.)

You can use the presentation "What If Part 1" as a guide through this lesson.

- Set a timer. Give students 1 minute to list all of the ways they have engaged in communication today (verbal/non-verbal).
- Have each team underline any communication methods that ALL team members wrote down.
- Using the Social Media Post Template handout, have students create a social media post that reflects their current status. Display these posts around the room.
- Present the scenario: what if all digital communication suddenly stopped working?
    - Encourage discussion. Have teams brainstorm a list of possible answers to the following question: "What will be impacted if digital communication is no longer an option?"
- How could you check the news to find out what caused the communication issue?
    - Point out that all digital devices would no longer work because they use digital communication internally between the processor and memory.
    - In your teams, create a definition of digital. (Remember, you don't have any digital devices to look it up!)
    - What kinds of communication will still work? (Hint: Not the TV: all of the signals are digital.)
    - Have teams try to build up a comprehensive list of the things we use that are digital.
    - Have teams complete the graphic organizer of what would be different in each of these places without digital communication:

At School	At Home	Other Places

Homework:

Students may not use any digital devices to complete this activity. This assignment must be handwritten. If students need a copy of their assignment for the class discussion, they must write another copy. Have students submit their assignment at the start of the next class.

1. Have the students interview three different people, outside of the class, using the following question:
    - How would life be different if we didn't have any means of digital communication?
2. Have the students write a summary of the interview that includes the following information:
    - Summary of the responses
    - Your opinion about the responses
    - What you learned by talking to others about the impact of losing digital communication

Session 2

Getting Started (5 min)

Have students write in their journals: What is the most important digital device in your life? Why is it the most important?

Activity (15 min) - Develop a Communication Plan

Use the presentation "What If Part 2" to remind the students about the scenario from the previous class. Working in the same teams from the previous class, have the students develop a step-by-step plan for getting a message to their parents without using any form of digital communication. This activity must be completed without using any digital tool.

Activity (30 min) - Discussing The Impact of Digital Communication

Teams brainstorm and organize ideas about the impact of digital communication. This activity uses the results from the Day 1 homework.

Discussion: Students work in their teams to answer the following questions:

1. How have the Web and the Internet changed the way people communicate and collaborate?
2. How does the impact of computing innovations differ between national and socioeconomic groups?
3. Describe a way in which social media has changed the way people communicate in the U.S.
4. In what ways have the Internet and the Web changed health care, access to information, entertainment, and online learning? How do these changes vary in different parts of the world?
5. Describe how two groups (e.g., in different geographic regions, from different cultural backgrounds, in different socioeconomic classes or different work industries) are impacted differently by social media and online access.
6. Describe how the impact of social media and online access differs in two different countries.
7. How does digitally enabled collaboration enhance human capabilities?

Homework

Each member of your team should choose one of the following topics:

- Science
- Art
- News
- Music
- Government
- Business

*Note – these topics are just suggestions

In the following question, fill in the blank with your chosen topic. Write a paragraph responding to the question. Be sure to include examples and evidence to support your ideas and answer.

- How does _____ depend on computers?

Session 3

Warm up / Activity 1 (35 min) - A Short Story

Imagine that the digital world that we know now never existed. There are no computers or cell phones -- no digital communication at all. Write a short story that takes place in this non-digital world. Include how your characters would communicate in different situations and how daily life would be. Be as creative as you can.

If time permits, have a few students share their stories.

Activity 2 (15 min) - Reflections

For this activity, students need to use their last social media post the created on Day 1.

Journal - Hooray! Digital communication has been restored after three years. Look at your last social media post, and think about the following questions.

1. How could a stranger interpret your last digital footprint? Was it positive, negative, or neutral?
2. How could you change your post to leave a more positive digital footprint?
3. What would be your first social media post now that digital communication has been restored?

Have students discuss their reflections with an elbow partner.

Options for Differentiated Instruction

Consider different ways to choose teams and assign team roles

Interview with a User of an Enhancing Technology

If you are familiar with an individual who benefits from an abilities-enhancing innovation or a technology that helps the individual overcome a disability, interview the person about the impact the technology has had on his or her life. Ask them questions about how the innovation works, how it has affected the way they live (the ways in which they play or work). Ask about how it has affected their family and friends. If possible, record the interview. Ask for permission to share with your classmates or to post online.

Speculate about Today's Innovations

Select a recent innovation - something recently in the news. Predict the impact that this innovation will have on individuals. Predict any societal impacts you can foresee. Label the impacts as positive or negative. Explain your reasons for the label.

Internal Use Only

Future Work

Should add assessments.

(http://csmatters.org) 1 - 2

0b1 - 0b10

Into the Light: How Computers and the Internet Enhance Innovation

Unit 1. Your Virtual World

Revision Date: Jul 02, 2016 (Version 2.0)
Duration: 3 50-minute sessions

Lesson Summary

Summary

Computing greatly affects the everyday lives of today's teens, but many of them are not consciously aware of these influences. In this lesson, students investigate the impact of the Internet on their lives.

Using a presentation about modern computers as an example, the teacher models the process of asking questions, organizing ideas, doing research, and giving a presentation. Students will then work in assigned groups to create presentations on the Internet and its Impact using online collaboration tools.

Students will experience some of the many collaborative tools available online.

Outcomes

- Students will describe multiple ways that the Internet impacts our lives.

- Students will use good research techniques to find and cite high-quality sources and to synthesize an original understanding of topics researched online.
- Students will define basic vocabulary about the Internet (cloud, server, etc.).
- Students will describe various tools that enable online collaboration.
- Students will collaborate using online tools to develop and refine a presentation.

Overview

1. Getting Started (5 min) – Students Think-Pair-Share about the size and scope of the Internet.
2. Teacher Overview (20 min) – Teachers present overview of the internet and introduce “pecha kuchas.”
3. Group Activity (30 min) – Student work in groups to organize ideas for research on a particular topic using mind maps.
4. Group Activity (65 min) – Students research their topic in groups and create “half pecha kuchas” to present to the class.
5. Presentations (20 min) – Each group presents their pecha kucha in the 3.5 min format.
6. Wrap Up (10 min) – Students discuss what they learned or observed from watching other groups’ presentations.

Learning Objectives

CSP Objectives

- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
  - 1.2.3A - Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
  - 1.2.3B - Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision.
  - 1.2.3C - Combining or modifying existing artifacts can show personal expression of ideas.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4B - Effective collaborative teams consider the use of online collaborative tools.
  - 1.2.4C - Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation.
  - 1.2.4D - Effective collaboration strategies enhance performance.
  - 1.2.4E - Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.
  - 1.2.4F - A collaboratively created computational artifact can reflect personal expressions of ideas.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
  - 7.1.1M - The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.
  - 7.1.1N - The Internet and the Web have changed many areas, including e-commerce, health care, access to information and entertainment, and online learning.
  - 7.1.1O - The Internet and the Web have impacted productivity, positively and negatively, in many areas.
- 7.1.2 - Explain how people participate in a problem solving process that scales.
  - 7.1.2D - Human capabilities are enhanced by digitally enabled collaboration.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
  - 7.3.1G - Privacy and security concerns arise in the development and use of computational systems and artifacts.
- 7.5.2 - Evaluate online and print sources for appropriateness and credibility.
  - 7.5.2B - Information from a source is considered relevant when it supports an appropriate claim or the purpose of the investigation.

NGSS Practices:

- 3. Planning and carrying out investigations
- 7. Engaging in argument from evidence
- 8. Obtaining, evaluation, and communicating information

NGSS Content:

- HS-ETS1-1. Analyze a major global challenge to specify qualitative and quantitative criteria and constraints for solutions that account for societal needs and wants.
- HS-ETS1-3. Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs that account for a range of constraints, including cost, safety, reliability, and aesthetics as well as possible social, cultural, and environmental impacts.

Key Concepts

Students should understand more about what the Internet is, what a computer is, and how the Internet affects our daily lives.

Students should develop an improved understanding of the power of the Internet as a positive agent of change.

Students should be able to use collaborative online tools to work in groups and use good research skills to deliver a worthwhile presentation.

Teacher Resources

Student computer usage for this lesson is: **required**

Teacher Resources

1. Unit 1 Lesson 2 LIGHT.pptx (overview of the lesson)
2. Modern Computerx.pptx (example of a short pecha kucha)
3. Script for Modern computers.docx
4. Bad presentation for contrast.pptx (example of how NOT to create a presentation)
5. Script for bad presentation.docx
6. Research and Collaboration Assessment.docx
7. Pecha Kucha student handout.docx
8. Pecha Kucha Assessment.docx

Student Resources:

1. Access to computers for using online collaboration resources.

Crowdsourcing Resources:

- Article on how crowdsourcing with GPS cellphone data might predict earthquakes: http://mashable.com/2015/04/10/smartphones-earthquake-early-warning-system/ (http://mashable.com/2015/04/10/smartphones-earthquake-early-warning-system/)

Lesson Plan

Session 1

Getting Started (5 min)

Journal

Instruct students to think about the following questions and journal about their thoughts. Afterwards, have them pair up and share their answers with each other. (See slide 2 of the presentation in the lesson folder: "Unit 1 lesson 2 LIGHT".)

- How big is the Internet?
- What kind of information is on the Internet?

Teacher Overview and Student Activity Introduction (20 minutes)

(Use the Unit 1 Lesson 2 LIGHT presentation for this activity.)

Part 1 - Teacher Presentation and discussion (10 min)

1. Use slides 3-4, and the students' ideas from the journaling exercises, to guide a class discussion on the Internet: what is out there and how does it affect our lives?
    - Try to get an idea of how much students know about what is on the Internet beyond email and social media.
    - If students need more idea starters, show the presentations on Communication Changes or Business Changes in the teacher folder.
    - Encourage students to continue to write down the things they know or could look up.
    - *Suggestion for follow-up activity:* Have students create "memes" and try to spread them through social networking to see the power of communication and collaboration.
2. Slides 5-7: Introduce the question of what a computer is. Then, using the script in the teacher folder, deliver the pecha kucha presentation on Modern Computers. Finally, discuss with the students what their assignment will be and give them some tips for creating their own presentation.
3. **Notes for teachers on presentation:**
    - "Pecha kucha" was originally used to refer to a specific talk format that contains 20 presentation slides, each of which are presented for exactly 20 seconds, using an auto-advance timer. The goal is to create a tight, effective presentation that moves quickly and conveys a lot of information in a short period of time.
    - For this exercise, you will be demonstrating a "half pecha kucha" that includes 10 slides -- presented at 20 seconds a slide, the presentation will last for 200 seconds (just over 3 minutes).
    - You will need to practice this presentation ahead of time, especially if you are not familiar with the "pecha kucha" format!

Part 2 - Introducing Student Activity (10 min)

1. In the remainder of the lesson, student groups will create their own "half pecha kuchas" on a topic about the Internet that they select.
2. Ask students what they learned:
    - About the content: modern computers.
    - About delivering a pecha kucha presentation: you need a script; you need to practice; it's automatically timed; you don't read from the screen.

3. Slides 8-10: Tell students they will be working for the next few days to create presentations about the Internet. Encourage some brainstorming of good questions to ask and write them down. (Slides 9-10 illustrate how this was done for the Modern Computer pecha kucha. Slide 11 includes some tips on doing good research. Lastly, slide 12 introduces the criteria that will be used to assess the presentation.)

- As you are doing this, give students the "Pecha Kucha Student Handout."
- ***Note:*** *If your students need more clarification on making a good presentation, you may want to deliver the "bad presentation for contrast" presentation (in the lesson folder, along with corresponding scripts) and have them describe why the presentation is bad.*

Group Activity: Collaboration and Organizing Ideas (30 minutes)

(See Research and Collaboration Assessment Rubric.)

Assign students to investigate and use online collaboration tools to:

- Collect the best questions on what they will research (google docs, Office 365, etc.).
- Create a mind map to organize the questions. MindMup (https://www.mindmup.com/) is recommended for creating the mind maps because it requires no signup. A list of tools is available here: http://elearningindustry.com/the-5-best-free-mind-mapping-tools-for-teachers (http://elearningindustry.com/the-5-best-free-mind-mapping-tools-for-teachers)
- Collect images from sources like pinterest, Google drive, or links in a document. (There are many other options.)

Session 2 - 3

Group Activity: Research and Presentation Preparation (65 minutes)

Part 1 - Preliminary research and collaborative development of pecha kuchas on the impact of the Internet (20 min)

Part 2 - Preliminary presentations/sharing and feedback (10 min)

Part 3 - Research and collaborative development of complete pecha kucha presentations on the impact of the Internet (35 min and homework / out-of-class time as desired)

Class Activity: Presentation of Research (20 min)

(See the self and group assessments and "Pecha Kucha Student Handout" in the lesson folder.)

- The timing for this session assumes that there are 4-5 group presentations, each in the pecha kucha 3.5-minute format.
- Every student should present some of the slides; the assessment includes both individual and group components.

Wrap up (10 min)

Share Student Learning

- Review knowledge gained from student presentations about facts learned, the research process, and presentation skills.

Options for Differentiated Instruction

- Suggestion: Have students create memes and try to spread them through social networking to see the power of communication and collaboration.

Evidence of Learning

Formative Assessment

Teachers are encouraged to have students present single slides and give each other feedback before continuing to do the research for the complete presentations.

Summative Assessment

Students create a presentation while working in groups and using online collaboration tools. A rubric will be used to assess the student group presentations along with self-reflections.

(http://csmatters.org) 1 - 3

0b1 - 0b11

Exploring Innovations

Unit 1. Your Virtual World

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Pre-lesson preparation

For better comprehension of the lesson, students should have ideally had experiences or have read about issues that have demonstrated how computing can be misused. This does not require assigned reading or review (just encourage them to watch the news and notice what is happening in the world), but you could have them bring in a current event article and summary of the event for homework as additional preparation.

Summary

Students will read about and discuss the issues that arise from the misuse of technology. Over the two sessions, these lessons will address social media, online retail and banking, cloud data storage, and government surveillance.

Outcomes

- Students will understand the consequences of Internet usage on personal privacy and security.
- Students will become aware of technologies designed to track their Internet usage.
- Students will understand the benefits and drawbacks of street cameras and facial recognition software.
- Students will understand both sides of the argument about government surveillance of electronic communications.

Overview

Session 1

1. Getting Started (5 min) – Journal about the effects of new technologies
2. Activity (40 min) – Jigsaw topics for research (working in topic groups)
3. Wrap up (5 min) – Discuss as a class what students have learned about each of the topics

Session 2

1. Getting Started (5 min) - Journal on government surveillance and data collection
2. Activity (10 min) - Finish jigsaw groups
3. Activity (30 min) - Regroup and share information gathered
4. Wrap up (5 min) - Select one topic to explore further

Learning Objectives

CSP Objectives

- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
- 3.1.1 - Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.
  - 3.1.1A - Computers are used in an iterative and interactive way when processing digital information to gain insight and knowledge.
- 3.2.1 - Extract information from data to discover and explain connections or trends
  - 3.2.1G - Metadata is data about data.
  - 3.2.1H - Metadata can be descriptive data about an image, a Web page, or other complex objects.
  - 3.2.1I - Metadata can increase the effective use of data or data sets by providing additional information about various aspects of that data.
- 3.2.2 - Determine how large data sets impact the use of computational processes to discover information and knowledge.
  - 3.2.2B - The storing, processing, and curating of large data sets is challenging.
  - 3.2.2D - Maintaining privacy of large data sets containing personal information can be challenging.
  - 3.2.2G - The effective use of large data sets requires computational solutions.

- 6.1.1 - Explain the abstractions in the Internet and how the Internet functions.
  - 6.1.1A - The Internet connects devices and networks all over the world.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
  - 7.1.1C - Social media continues to evolve and fosters new ways to communicate.
  - 7.1.1D - Cloud computing fosters new ways to communicate and collaborate.
  - 7.1.1H - Social media, such as blogs and Twitter, have enhanced dissemination.
  - 7.1.1M - The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.
  - 7.1.1N - The Internet and the Web have changed many areas, including e-commerce, health care, access to information and entertainment, and online learning.
  - 7.1.1O - The Internet and the Web have impacted productivity, positively and negatively, in many areas.
- 7.1.2 - Explain how people participate in a problem solving process that scales.
  - 7.1.2G - The move from desktop computers to a proliferation of always-on mobile computers is leading to new applications.
- 7.2.1 - Explain how computing has impacted innovations in other fields.
  - 7.2.1C - Computing enables innovation by providing the ability to access and share information.
  - 7.2.1G - Advances in computing as an enabling technology have generated and increased the creativity in other fields.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
  - 7.3.1A - Innovations enabled by computing raise legal and ethical concerns.
  - 7.3.1D - Both authenticated and anonymous access to digital information raise legal and ethical concerns.
  - 7.3.1E - Commercial and governmental censorship of digital information raise legal and ethical concerns.
  - 7.3.1G - Privacy and security concerns arise in the development and use of computational systems and artifacts.
  - 7.3.1H - Aggregation of information, such as geolocation, cookies, and browsing history, raises privacy and security concerns.
  - 7.3.1I - Anonymity in online interactions can be enabled through the use of online anonymity software and proxy servers.
  - 7.3.1J - Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.
  - 7.3.1L - Commercial and governmental curation of information may be exploited if privacy and other protections are ignored.

Essential Questions

- What are some potential beneficial and harmful effects of computing?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

Teacher Resources

Student computer usage for this lesson is: **required**

These materials may be useful if you want to spend some time with the entire group discussing a few key topics.

- Video: "Your Facebook life in 62 Seconds" http://www.cnn.com/2014/02/04/tech/social-media/facebook-look-back-video/ (http://www.cnn.com/2014/02/04/tech/social-media/facebook-look-back-video/)
- Article on Facebook about privacy: https://www.facebook.com/about/privacy/ (https://www.facebook.com/about/privacy/) .
- Article: Ten Commandments of Computer Ethics - http://en.wikipedia.org/wiki/Ten_Commandments_of_Computer_Ethics (http://en.wikipedia.org/wiki/Ten_Commandments_of_Computer_Ethics)

Lesson Plan

Session 1

Getting Started (5 min) - Journal

The purpose of this session is to make students think about the different ways in which they as individuals, and society as a whole, are more vulnerable because of new technologies.

Students should consider the following question and record their reflections in their journals:

- With the invention of new technologies, what additional risks do we face (personally and societally)?

Guided Activity (40 min)

For this activity, teachers will use "Jigsaw Groups":

1. Create student groups with 4 students per group. Each student will select one topic (innovation or aspect of technology).
2. Redistribute class groups by topic. Each topic group will work together to explore resources and take notes during the 40-minute time block.
   - Groups should use the worksheet (ExploringInnovationsWorksheet.docx) to identify and record potential impacts of the technology, whether they primarily affect individuals or society as a whole, whether they are positive or negative, evidence of that impact, and the source they used to find the information. (Each student in the group should make their own copy of the worksheet, so they can bring them back to their original jigsaw groups.)

- After completing the worksheet, students should complete the Venn diagram (ExploringInnovationVenn.docx) to summarize the key impacts of an innovation, and who is affected.

3. The students in the topic groups will report back to their original group of 4 students in session 2, thus completing the "jigsaw."

The topics (and examples of positive (+) and negative (-) impacts) include:

- Social media (+ connecting at a distance, - cyberbullying)
- Online retail, banking, and businesses (+ convenience, - identity theft)
- Cloud data storage (+ information sharing, - loss of privacy)
- Government surveillance (+ find terrorist threats, - loss of privacy)

For each of the above topics, there is a resource sheet in the lesson folder that can be provided to student groups. (Optionally, you may want to create additional resource sheets, or let students select other topics and find their own resources.)

Wrap Up (5 min)

As a class, review each of the larger topic areas and remind students that they will be sharing information in their original groups the next day. If students need too much additional time to research the topic, you may consider assigning them to complete the research independently for homework.

Session 2

Getting Started (5 min)

Students should take a few minutes to journal about the following prompt:

- Think about your typical day. How often do you think that your image has been captured by a surveillance camera? List all of the places where your image may have been captured. Also, consider what you have done in the past week. What data might have been collected about you somewhere over the past week?

Guided Activity (10 min)

Topic Groups: Have students briefly assemble into topic groups to compare notes.

Guided Activity (30 min)

- Jigsaw Groups: Have students assemble into their original jigsaw groups. Each member will present the information on the topic that was researched. All notes need to be shared within these groups.
- You may regroup and discuss the topics as a class if time permits.

Wrap Up (5 min)

Each student should select a topic that they would like to explore further and write the topic in their journal. It might be a narrow subtopic from the broader topics that were explored within this lesson. They might also want to write down a few interesting innovations connected to a topic. They will refer back to this during the practice performance lesson later in the unit.

Internal Use Only

Pending Tasks

- Handout: "Your Facebook in 62 Seconds.pdf" (in the Lesson Resources folder) - ***NOTE: Do not distribute this handout until we have verified that we have permission to reuse it.***

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary

A bit is a single unit of information. Bits are the fundamental building blocks of digital computing. There are many different ways to represent a single bit physically, and collections of bits can be combined to represent everything from numbers, to electronic books, to control programs for interstellar probes. In this lesson, students will learn how bits are stored and how they can be used to represent information. Students will further explore how numbers can be represented in binary form, how to convert numbers between these different forms, and how they are used by different applications.

Outcomes

- Understand the abstraction of a "bit" and how bits can be used to represent different kinds of information.
- Be able to convert numbers between decimal, binary, and (optionally) hexadecimal forms.
- Be able to identify formats used to represent numbers, pictures, audio, and video data.
- Understand how these forms of bit representation are used in modern technology.

Overview

Session One

1. Getting Started (5 min) - Jacquard loom
2. Guided Activity (10 min) - Light bulb abstraction of bits
3. Interactive Teacher Presentation (25 min) - "It's Just Bits"
4. Wrap Up (10 min) - Binary conversions worksheet

Session Two

1. Getting Started (5 min) - Journal and completing worksheet
2. Guided Activity (20 min) - Conversions (two options: Cisco Binary Game or hexadecimal conversions)
3. Real-World Connections (20 min)
4. Wrap-up (5 min) - Journal

Optional Activities

- Homework - Generate conversion worksheets using http://www.worksheetworks.com/math/numbers/systems.html (http://www.worksheetworks.com/math/numbers/systems.html)
- (30 min) Web-quest to explore different ways of physically representing bits.
- Four research activities on:
  - Charles Babbage and Ada Lovelace
  - Hollerith's Tabulation Machine
  - Qbits
  - Weaving with the Jacquard Loom

Source

Parts of this lesson were adapted from code.org.

Learning Objectives

CSP Objectives

- 2.1.1 - Describe the variety of abstractions used to represent data.
  - 2.1.1A - Digital data is represented by abstractions at different levels.
  - 2.1.1B - At the lowest level, all digital data are represented by bits.
  - 2.1.1C - At a higher level, bits are grouped to represent abstractions, including but not limited to numbers, characters, and color.
  - 2.1.1D - Number bases, including binary, decimal, and hexadecimal, are used to represent and investigate digital data.
  - 2.1.1E - At one of the lowest levels of abstraction, digital data is represented in binary (base 2) using only combinations of the digits zero and one.
  - 2.1.1F - Hexadecimal (base 16) is used to represent digital data because hexadecimal representation uses fewer digits than binary.
- 2.1.2 - Explain how binary sequences are used to represent digital data.
- 4.1.1 - Develop an algorithm for implementation in a program.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
  - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
  - 5.5.1B - Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.
  - 5.5.1C - Real numbers are approximated by floating-point representations that do not necessarily have infinite precision.

2.1.1 ABDE

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP4: Model with mathematics.
- MP8: Look for and express regularity in repeated reasoning.

Common Core Math:

- N-Q.1-3: Reason quantitatively and use units to solve problems

Common Core ELA:

- RST 12.1 - Cite specific textual evidence to support analysis of science and technical texts, attending to important distinctions the author makes and to any gaps or inconsistencies in the account.
- RST 12.2 - Determine central ideas and conclusions in the text
- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.7 - Integrate and evaluate multiple sources of information presented in diverse formats and media
- RST 12.10 - Read and comprehend science/technical texts
- WHST 12.1 - Write arguments on discipline specific content
- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.8 - Gather relevant information from multiple authoritative print and digital sources, using advanced searches effectively; assess the strengths and limitations of each source
- WHST 12.9 - Draw evidence from informational texts to support analysis, reflection, and research

NGSS Practices:

- 5. Using mathematics and computational thinking

Key Concepts

The students will...

- Understand the abstraction of a "bit" and how bits can be used to represent different kinds of information.
- Be able to convert numbers between decimal, binary, and (optionally) hexadecimal forms.
- Be able to identify formats used to represent numbers, pictures, audio, and video data.
- Understand how these forms of bit representation are used in modern technology.

Essential Questions

- How can computing and the use of computational tools foster creative expression?
- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How can computational models and simulations help generate new understanding and knowledge?
- How are number values converted across decimal, binary, and hexadecimal representations?
- How are digital colors represented with hexadecimal codes?
- What is the algorithm to convert a number to or from hexadecimal with another number system?

Teacher Resources

Student computer usage for this lesson is: **required**

For the Student

- Journal
- Option: Access to a web browser with Flash to play the Cisco Binary Game: http://forums.cisco.com/CertCom/game/binary_game_page.htm (http://forums.cisco.com/CertCom/game/binary_game_page.htm)
- Option: Access to a web browser and collaborative file creation/sharing site, such as Google Documents, to complete a web quest

For the Teacher

- Optional: punch cards of any kind
- Access to a web browser to show YouTube videos:
  - Jacquard Loom video (in Getting Started): http://www.youtube.com/watch?v=lwozgRPLVC8 (http://www.youtube.com/watch?v=lwozgRPLVC8)

    - Octopus Counting video by Michael Littman (in "JustBits" presentation): https://www.youtube.com/watch?v=4p-9-nK-mwY (https://www.youtube.com/watch?v=4p-9-nK-mwY)
- Optional: Create a document (Representing Bits) in the file sharing site and make it accessible to all of the students. Add four headings (Punch cards, Magnetic polarity, Electrical voltage, Light intensity) under which students will add content in the collaborative research activity.
- Optional: Additional conversion worksheets generated at http://www.worksheetworks.com/math/numbers/systems.html (http://www.worksheetworks.com/math/numbers/systems.html)

In the Lesson Resources folder:

- Presentation: JustBits
- BinaryConversionWorksheet
    - Give these directions to students during the lesson
- AdditionalResource_bin2dec
- AdditionalResource_BinaryWorksheet
- AdditionalResource_binmagic
- For Lecture:
    - HexBinaryExamples
    - HexDecimalExamples
- Hexadecimal worksheets:
    - Worksheet: HexConversionsWorksheet
    - Homework: HexConversionsHomework
- Answer Keys:
    - HexConversionsWorksheetKEY
    - HexConversionsHomeworkKEY

Available on the Web for Teachers:

- Binary, Decimal, Hexadecimal Conversion Chart
    - http://tonystrains.com/download/DCC_DecBiHex_Chart.pdf (http://tonystrains.com/download/DCC_DecBiHex_Chart.pdf) (http://tonystrains.com/download/DCC_DecBiHex_Chart.pdf)

Lesson Plan

Session One

Getting Started (5 min)

Jacquard loom: early computer programming

The teacher will introduce the "Jacquard Loom," an early machine that made use of punch cards to make complicated textiles.

- View the video: http://www.youtube.com/watch?v=lwozgRPLVC8 (http://www.youtube.com/watch?v=lwozgRPLVC8)
- If you have punch cards, pass them around while students are watching the video.

Guided Activity (10 min)

Represent Values as a Light Switch

1. Instruct students to answer these questions with an elbow partner:
    - Using a single light switch, how can you represent the value 0? 1?
    - If you have two light switches (connected to two different lights in the same room), how many values can you represent? Describe how you would represent each value (numbers, symbols, etc.).
2. Invite students to share their ideas with the class.
3. Engage the entire class with the following questions. Show the various combinations.
    - If I add a third switch, how many values can I represent?
    - How many values can I represent with four switches? Do you notice a pattern?

Teaching Tip: Guiding the students toward understanding that the number of switches determines how many numbers can be represented. The pattern is 1 switch = 2 (or 2^1) numbers, 2 switches = 4 (or 2^2), 3 switches = 8 (or 2^3), etc.

Interactive Teacher Presentation (25 min)

Use the "JustBits" presentation in the lesson folder to explore different ways to represent bits, and different ways in which bits can be used to store different kinds of information.

General Presenting Tips:

- Throughout the presentation, the slides have questions on them that the students should be thinking about. Try to ensure that students are actively considering these questions and that they understand how to answer them before moving on.
- You may wish to call on students randomly, have everyone write down an answer, or have students suggest different answers that you write on the board and then have the class work on. (**Note:** Incorrect answers are a great opportunity to diagnose and remedy conceptual errors. Remember, there are no "bad answers," just ones that reveal different levels of understanding.)

Presentation Guide:

- As you go through the slides, discuss how the different uses of bits relates to the representation of bits. Eight bits can be interpreted in many ways: as an unsigned integer from 0 to 255, as a signed integer from -127 to 128, as a single ASCII character, or as a red, green, or blue color value associated with a pixel in a screen display.
- Emphasize to the students that a "bit" is just an abstraction. It refers to the value of a single piece of information stored in an "on/off", "zero/one", or "yes/no" format. Similarly, any given interpretation a collection of bits as information is an abstraction that lets us move from "a bunch of bits" to "data and information."
- **Slide 5** (Storing Bits): *If you are planning to do the optional web quest to explore different bit representations, you could encourage students to think about different ways that bit could be stored, to get them thinking about that question.*
- **Slide 11** (Conversions): This is a fairly quick introduction to binary/decimal conversions. You may wish to spend some time at the board working through more examples. Students will have an opportunity later in the class to use the "Cisco Binary Game" (or optionally paper worksheets) to practice binary/decimal conversion.
- **Slide 12** (Hexadecimal): *If you are planning to have the students work on hexadecimal number conversions, you may wish to spend a bit of extra time on this slide to make sure they understand the concept more clearly.*
- **Practice Problems, Slide 18:** Give the students a few minutes to start working on the first problem, then lead the class in a brief discussion to make sure they're approaching it correctly.
  - They should be multiplying the number of pixels (600 * 400 = 240,000) by the number of bytes (3 bytes in a pixel) by the number of bits in a bite (8 bits in a byte).
  - The total number of bits = (240,000 * 3 * 8) = 5,760,000 bits.
  - CD answer: 44,100 samples/second * 16 bits/sample = 705,600 bits/second.
  - One 3-minute song = 705,600 bits/second * 60 seconds/minute * 3 = 127,008,000 bits.

Wrap Up (10 min)

Converting between decimal and binary

Use the "BinaryConversionWorksheet" in the lesson folder (from Code.org) to let students explore (individually or in pairs/small groups) how to build an algorithm for converting binary to decimal. As you progress through the activity, answer questions as they arise, or ask students to explain how they arrived at the answer to check for understanding.

Other extensions and activities that may be useful:

- List the numbers from 0 to 15. Show students how they are represented in binary. Practice representing the grade level of various students in binary.
- Show an algorithm for converting binary to decimal.
- Show an algorithm for converting decimal to binary.
- Convert an IP-V4 address from decimal to binary.

Session Two

Getting Started (5 min)

Introduction

- Have students describe what they know about representing numbers in binary code from the previous lesson (journal or discussion). (You may want to put a few conversion problems on the board as an entry check.)
- Complete any remaining problems in the "Student Activity Guide: Converting Decimal to Binary."

Guided Activity: Conversions (20 min)

Option 1:

- Have the students play the Cisco Binary Game. http://forums.cisco.com/CertCom/game/binary_game_page.htm (http://forums.cisco.com/CertCom/game/binary_game_page.htm) You could also have the students teach a friend how to play the game, or create a quiz to determine if the friend has learned the basics of the binary number system. To ensure progress, you could have the students show you and/or a partner when they have reached a particular level of the game.

Option 2:

- If your students are comfortable with binary/decimal conversions, you could have them also master hexadecimal conversions. (See "HexBinaryExample"s and "HexDecimalExamples" in the Lesson Resources folder for specific directions and suggested examples and/or the PowerPoint slides in the lesson folder.)

- You may want to add additional examples or have students come to the board to show their work as you go through the different types of conversions. (See the "HexConversionWorksheet" for distribution of those additional practice problems done during class.)

Extension: If you want to give your students more practice problems for in-class practice or homework, you can use the worksheet generator at http://www.worksheetworks.com/math/numbers/systems.html (http://www.worksheetworks.com/math/numbers/systems.html)

Real World Connections (20 min)

How are text, colors and images saved in hexadecimal format? Select some of the following to present and discuss with the class.

1. How is text converted to hexadecimal format?
   Ascii to Hexadecimal conversion chart (https://drive.google.com/open?id=1NTLX9O_GaI_WmDSiMSmy21HwHNbKhUDCigL-Na1Q-l4)
   Text to hexadecimal conversion tool - http://www.string-functions.com/hex-string.aspx

   (http://www.string-functions.com/hex-string.aspx)
2. Where are hexadecimal numbers used to represent digital data? Colors!
   Color Conversion Website: http://rapidtables.com/convert/color/index.htm

   (http://rapidtables.com/convert/color/index.htm)
3. How are picture files saved with hexadecimal format?
   http://www.colorcodepicker.com/

   (http://cs10kcommunity.org/forum/lessons-share-encourage-active-learning)
4. Color Chart of 216 Web-safe Colors in hexadecimal form - http://www.w3schools.com/html/html_colors.asp
   (http://www.w3schools.com/html/html_colors.asp)

Wrap-Up (5 min)

Journal

Students should consider the following prompt, and record their thoughts in their journals:

- Why is the abstract idea of an image stored in a jpeg file useful even though it hides important details about the actual file format? When a user takes a digital photograph, they capture a digital image in their camera. Assuming the pictures are in the jpeg format, explain the difference between the abstract idea of an image and what is actually stored by the camera.
- **Alternative question:** Now that you have seen hexadecimal, what other number systems might be useful in computer science? Give advantages and disadvantages of each.

Optional Activities

Activity 1 - Collaborative Web-Quest (20 min)

Part 1 - Exploring Physical Bit Representations

- Group students into small teams. Assign each team to research how computers represent bits with one of the following:
  - Punch cards
  - Magnetic polarity
  - Electrical voltage
  - Light intensity
- Students can record their findings in a common file such as a Google doc, so that all students can edit and see the combined results of the research. Encourage students to create and/or add images that more completely illustrate what they have learned.

Teaching note: The research and writing activity in this lesson presents an opportunity to talk about copyright laws and emphasize that copied content must be credited to the rightful author or organization.

Part 2 - Sharing the research

- Allow students in each group to briefly share the most important details about what they learned about their assigned topics. If more than one group researched the same topic, they can report to the class as one team.
- Encourage the class to ask questions.
- Ask students if they know of any additional ways to represent bits.

Teaching note: Model writing skills through a variety of writing opportunities and prompts. Encourage students to write complete sentences that clearly communicate their ideas.

Activity 2 - Charles Babbage and Ada Lovelace

- Have each student research the machines designed and programmed by Charles Babbage and Ada Lovelace. How did these machines represent bits? Investigate Babbage's and Lovelace's intellectual roles in creating their designs.
- Communicate your learning in any effective format. Consider images, a collage, a recording, a screen play script, or a compare-and-contrast chart of the contributions to computer science of both Babbage and Lovelace.
- Student should prepare to discuss their findings with the class.

Activity 3 - Hollerith's Tabulation Machine

- Have each student research the machines that were designed by Herman Hollerith. How did these machines represent bits? What was the purpose of the machines? How did his device change history?
- Predict how history might have been different if Hollerith hadn't created this device.
- Student should prepare to discuss their findings with the class.

Activity 4 - Learn about Qbits

- Have each student research "Qbit" and how it is used in quantum computing. Be particularly attentive to the vocabulary used to describe this technology. What implications does it have for the future of computing? Report your findings in any effective format.
- Student should prepare to discuss their findings with the class.

Activity 5 - Weaving

- Have each student create a small weaving project by following the directions shown in the video about the Jacquard loom:
  - http://www.instructables.com/id/How-to-Weave-on-a-Frame-Loom/step2/Warp/ (http://www.instructables.com/id/How-to-Weave-on-a-Frame-Loom/step2/Warp/)
- Students respond to this prompt in their journals:
  - How did the invention of the Jacquard weaving loom change society? (consider technologically, economically, socially) Compare these changes to those brought by a recent innovation in computing technology. Be sure to speculate on the long-term impact and significance of these inventions.

Options for Differentiated Instruction

The Extension section above gives a variety of outside activities, some of which are appropriate for verbal and others for tactile learners.

1. Instead of flashlights, the same task can be done with large cards that are black on one side and white on the other.
2. If you have the space for a physical activity, have two (or more) teams (of 4 or 5 students each) line up.
   - Each student represents a bit, with the student on one end being the bit in the 1's place, the next student representing the 2's place, the next the 4's place, etc.
   - The students start in a standing position, which represents neither 1 nor 0. To represent a 1, the student's arms must be stretched straight overhead; to represent a 0, the student must squat down.
   - You then call a number (one that can be represented using that many bits). The two teams then race to get their team to represent that number. The first team to have it correct gets a point. They normalize (all stand with no arms up) and a new number is called.
   - Ask them about patterns they find during the game. (The 1's place student should notice that if the number is odd, their arms are up, but if the number is even, they are squatting. The student representing the highest bit should notice that their arms are up if the number is larger than or equal to their place value.)

Note: If there are students in the class with physical limitations who are unable to stand, stretch, or squat, the game can be modified appropriately:

- Have the students play the game seated, with a card in their lap. (It represents neither a 1 nor a 0 when in their lap.) They will hold up a black side (to represent 0) or a white side (to represent 1) when a number is called.

Visual - Decimal, Binary, Octal, and Hexadecimal Number Systems Video - http://whyu.org/whyUPlayer.php?currentchapter=3&currentbook=1&youtubeid=5sS7w-CMHkU (http://whyu.org/whyUPlayer.php?currentchapter=3&currentbook=1&youtubeid=5sS7w-CMHkU)

OR - https://www.youtube.com/watch?v=_oaBT-TndCs (https://www.youtube.com/watch?v=_oaBT-TndCs) **(https://www.youtube.com/watch?v=_oaBT-TndCs)**

Kinesthetic - Hexadecimal Drum Machine - h (http://whyu.org/whyUPlayer.php?currentchapter=3&currentbook=1&youtubeid=5sS7w-CMHkU)ttp://www.mathsisfun.com/games/hex-drums.html (http://www.mathsisfun.com/games/hex-drums.html)

Auditory - Hexadecimal File Music - https://www.youtube.com/watch?v=fyBf4Y2mVzs (https://www.youtube.com/watch?v=fyBf4Y2mVzs)

Extension Activity 1. Students that have mastered the conversion techniques can peer-tutor students [one-on-one] that are having difficulty solving the conversions.

Extension Activity 2. Students can observe their computer's network interface card (NIC) MAC address in hexadecimal, and convert the MAC address to binary and decimal.

- To find your computer's network interface card (NIC) instructions: http://www.miami.edu/index.php/telecommunications/student_services/student_technical_support/find_your_mac_address/ (http://www.miami.edu/index.php/telecommunications/student_services/student_technical_support/find_your_mac_address/)

See algorithms of number system conversions:

- Hexadecimal to Decimal Number Systems Converter in Excel to see algorithm - http://www.schooltube.com/video/c98baf4e105241b283d4/Bin2Dec2Hex%20-%20Lesson%206%20-%20Converting%20a%20hexadecimal%20number%20to%20decimal

(http://www.schooltube.com/video/c98baf4e105241b283d4/Bin2Dec2Hex%20-%20Lesson%206%20-%20Converting%20a%20hexadecimal%20number%20to%20decimal)

- Hexadecimal to Binary Number Systems Converter in Excel to see algorithm - http://www.wikihow.com/Sample/Hexadecimal-to-Binary-Converter (http://www.wikihow.com/Sample/Hexadecimal-to-Binary-Converter)

- http://www.schooltube.com/video/04a13760ada644a2a7d4/Bin2Dec2Hex%20-%20Lesson%203%20-%20Converting%20a%20hexadecimal%20number%20to%20binary (http://www.schooltube.com/video/04a13760ada644a2a7d4/Bin2Dec2Hex%20-%20Lesson%203%20-%20Converting%20a%20hexadecimal%20number%20to%20binary)

Alternative presentation for Gifted Students doing Decimal to Hexadecimal Conversion:

For decimal number x:

1. Get the highest power of 16 that is less than the decimal number x:
2. $16^n < x$, (n=1,2,3,...)
3. The high hex digit is equal to the integer if the decimal number x divided by the highest power of 16 that is smaller than x:
4. $d_n = \text{int}(x / 16^n)$
5. Calculate the difference Δ of the number x and the hex digit dn times the power of 16, 16n:
6. $\Delta = x - d_n \times 16^n$
7. Repeat step #1 with the difference result until the result is 0.

Example

Convert x=603 to hex:

n=2, 16^2=256 < 603

n=3, 16^3=4096 > 603

So

$n = 2$

$d_2 = \text{int}(603 / 16^2) = 2$

$\Delta = 603 - 2 \times 16^2 = 91$

$n = 1$, $x = \Delta = 91$

$d_1 = \text{int}(91 / 16^1) = 5$

$\Delta = 91 - 5 \times 16^1 = 11$

$n = 0$, $x = \Delta = 11$

$d0 = \text{int}(11 / 16^0) = 11_{10} = B_{16}$

$\Delta = 11 - 11 \times 16^0 = 0$

$(d_2 d_1 d_0) = 25B$

Answer: $x = 603_{10} = 25B_{16}$

Learn to Count in Binary and Hexadecimal - http://webelfin.com/webelfindesign/counthex.html (http://webelfin.com/webelfindesign/counthex.html)

ADDITIONAL/ALTERNATE FORMATIVE ASSESSMENT MATERIAL or Class Activity if time permits - from unplugged.com

http://csunplugged.org/binary-numbers (http://csunplugged.org/binary-numbers)

Brainstorm reasons for storing and communicating secret messages. Challenge students to think of both helpful and problematic reasons. Students record at least two of each.

View steganography presentation. Presentations are available on YouTube.

- Abdullah Seddiq (MIT Blossoms) (http://blossoms.mit.edu/home)has Counting Systems (http://blossoms.mit.edu/videos/lessons/counting_systems)with teacher's guides and additional resources. This video aims to explain counting systems (Decimal, Binary, Hexadecimal). Students will get to know how to convert numbers between these systems. Also students will learn how to do some byte and bit level operations. They will use a Visual Basic (VB) application

that changes colors through logical operation on numbers. See also The Magic Picture: Steganography in Bitmap Files (http://blossoms.mit.edu/videos/lessons/magic_picture_steganography_bitmap_files)

- Video - http://blossoms.mit.edu/videos/lessons/magic_picture_steganography_bitmap_files (http://blossoms.mit.edu/videos/lessons/magic_picture_steganography_bitmap_files)

Evidence of Learning

Formative Assessment

Assessment Questions:

- Describe the pattern for even and odd numbers that you see in the binary number system. Speculate on why this pattern exists.
- Why is the binary number system used in computers?
- Why is the Jacquard loom important in the history of computing?
- How are bits represented in punch cards?
- How are bits represented magnetically?
- How are bits represented with voltage?
- How are bits represented with light?
- Are students correctly answering questions in the Getting Started activity: Represent values with a light switch? On the Student Activity Guide: Converting Decimal to Binary?
- Decimal, binary, and hexadecimal value conversions:
  - Write the step-by-step process needed for the following number conversions:
    - hexadecimal to binary
    - binary to hexadecimal
    - hexadecimal to decimal
    - decimal to hexadecimal
  - Assign a sequence of consecutive hexadecimal numbers - one to each person. After they have converted them to decimal and to binary, have them compare to find patterns in their answers.
  - Distribute worksheets so all students practice conversions of all types. See "BinaryConversionWorksheet" and "HexConversionWorksheet"

Summative Assessment

Journal check - questions presented as described in the lesson plan

ASSESSMENT QUESTIONS

- Why is the binary number system used in computers?
- Describe the pattern you observe for even and odd numbers in binary. Why do you think this pattern emerges?
- What other patterns do you see in binary numbers?
- Convert the ages of your family members to binary. How many bits are needed to express the ages?
- Describe an algorithm for converting decimal numbers to binary.
- Describe an algorithm for converting binary numbers to decimal.

Getting Started section: questions on converting between bases and describing the purpose of hexadecimal numbers.

(http://csmatters.org) 1 - 5

0b1 - 0b101

How Innovation Affects Our Lives

Unit 1. Your Virtual World

Revision Date: Aug 23, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

Computing innovations have the potential to significantly impact our lives, both positively and negatively. In order to understand the full range of impact (or lack of impact) of a given innovation, one must consider how differences in geographic location, culture, and socioeconomic status influence the effect that given innovation has on a specific group of people. Students learn about the digital divide on national and global levels and analyze how three different computing innovations impact people, making ethical considerations while doing so in order to determine if the impact is beneficial or harmful.

Outcomes

Students will:

- Discuss the ethical concerns that arise from a given computing innovation.
- Describe the ways in which the digital divide affects individuals and groups of people.
- Analyze the impact of a given computing innovation with respect to a specific group of people and categorizing the impact as beneficial or harmful.

Overview

1. Getting Started (5 min)
2. Guided Presentation (20 min)
3. Independent Analysis (20 min)
4. Wrap-up (5 min)

Learning Objectives

CSP Objectives

- 7.1.2 - Explain how people participate in a problem solving process that scales.
  - 7.1.2G - The move from desktop computers to a proliferation of always-on mobile computers is leading to new applications.
- 7.2.1 - Explain how computing has impacted innovations in other fields.
  - 7.2.1A - Machine learning and data mining have enabled innovation in medicine, business, and science.
  - 7.2.1G - Advances in computing as an enabling technology have generated and increased the creativity in other fields.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
  - 7.3.1A - Innovations enabled by computing raise legal and ethical concerns.
  - 7.3.1B - Commercial access to music and movie downloads and streaming raises legal and ethical concerns.
  - 7.3.1C - Access to digital content via peer-to-peer networks raises legal and ethical concerns.
  - 7.3.1F - Open source and licensing of software and content raise legal and ethical concerns.
  - 7.3.1P - The Digital Millennium Copyright Act (DMCA) has been a benefit and a challenge in making copyrighted digital material widely available.
- 7.4.1 - Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts.
  - 7.4.1A - The innovation and impact of social media and online access varies in different countries and in different socioeconomic groups.
  - 7.4.1B - Mobile, wireless, and networked computing have an impact on innovation throughout the world.
  - 7.4.1C - The global distribution of computing resources raises issues of equity, access, and power.
  - 7.4.1D - Groups and individuals are affected by the "digital divide" - differing access to computing and the Internet based on socioeconomic or geographic characteristics.
  - 7.4.1E - Networks and infrastructure are supported by both commercial and governmental initiatives.

Math Common Core Practice:

- MP3: Construct viable arguments and critique the reasoning of others.

Common Core ELA:

- RST 12.1 - Cite specific textual evidence to support analysis of science and technical texts, attending to important distinctions the author makes and to any gaps or inconsistencies in the account.
- RST 12.2 - Determine central ideas and conclusions in the text
- RST 12.6 - Analyze the author's purpose in providing an explanation, describing a procedure
- WHST 12.1 - Write arguments on discipline specific content
- WHST 12.4 - Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience

NGSS Practices:

- 1. Asking questions (for science) and defining problems (for engineering)
- 7. Engaging in argument from evidence

Key Concepts

- Innovations have positive and negative impacts.
- The digital divide can significantly influence the impact and utility of a given innovation for different groups of people.
- In order to understand the impact of a given innovation, it is necessary to consider if and how the innovation affects different groups of people in the world and whether that impact is beneficial or harmful to that group.

Essential Questions

- What are some potential beneficial and harmful effects of computing?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

Teacher Resources

Student computer usage for this lesson is: **optional**

For the Student

- Journal
- Copy of Find My Friends App article (if not reading online): http://www.news.com.au/technology/gadgets/these-smartphone-users-share-how-tracking-app-find-my-friend-has-saved-them/story-fn6vihic-1226752063613 (http://www.news.com.au/technology/gadgets/these-smartphone-users-share-how-tracking-app-find-my-friend-has-saved-them/story-fn6vihic-1226752063613)
- Copy of Analyzing Impacts of Digital Innovations worksheet (Lesson1_5.docx in lesson resources folder)
- Copy of Lesson 1.5 Homework worksheet (Lesson1_5hw.docx in lesson resources folder)

For the Teacher

- Access to a web browser to show YouTube videos:
  - Is Developing Artificial Intelligence (AI) Ethical? | Idea Channel | PBS Digital Studios: https://www.youtube.com/watch?v=95KhuSbYJGE (https://www.youtube.com/watch?v=95KhuSbYJGE)
  - PBS News Hour segment about the digital divide: https://www.youtube.com/watch?v=X537MiN6COI (https://www.youtube.com/watch?v=X537MiN6COI)
- Access to a web browser to pull up links:
  - https://www.youtube.com/watch?v=7AF18DUIH1Y (https://www.youtube.com/watch?v=7AF18DUIH1Y)
  - http://computer.howstuffworks.com/napster.htm (http://computer.howstuffworks.com/napster.htm)http://www.internetlivestats.com/internet-users-by-country/ (http://www.internetlivestats.com/internet-users-by-country/)
  - http://data.worldbank.org/indicator/IT.NET.USER.P2?view=map (http://data.worldbank.org/indicator/IT.NET.USER.P2?view=map)
  - http://www.bbc.com/news/technology-34780127 (http://www.bbc.com/news/technology-34780127)

In the Lesson Resources folder:

- Presentation (Lesson1_5)
- Analyzing Impacts of Digital Innovations Worksheet (Lesson1_5wkst)
- Homework (Lesson1_5hw)

Lesson Plan

Getting Started (5 min)

Present "Lesson1_5" PowerPoint slides (in Lesson Resources folder).

- **Slide 2:** Students will respond to the following prompt in their journals: List three questions you could ask to decide if an innovation is "ethical".
- When the students finish, have them discuss their answers. *(Possible answers: does it cause physical, emotional, cultural, environmental, economic or social harm?)*

Guided Presentation (20 min)

Analyzing the Ethics of an Innovation (Slides 3 – 10) :

- **Slide 3:** Show students the definition of “ethics”.
- **Slide 4:** Have students vote as to whether or not “ethical” and “legal” are the same. This is especially relevant with respect to innovations because laws cannot be made about a specific technology before it exists. I.e. laws governing the use of a specific technology must be made after its emergence.

- **Slide 5:** Present the necessary considerations for analyzing the impacts of an innovation, focusing on the first point about whether or not a given impact is negative or positive.
- **Slides 6-9:** Present the background information on Napster and the legal proceedings that resulted from copyright infringement issues. It is important to establish the state of online music sharing/streaming in the late 1990's when Napster emerged so that students understand the controversy surrounding its creation and implementation.
- **Slide 10:** Have students analyze whether or not they think Napster was an ethical innovation by writing down the beneficial and harmful impacts of its creation and use. **Slide 11** provides some possible answers to the prompt.

The Digital Divide (Slides 12 - 17):

- **Slide 12:** Revisit the necessary considerations for analyzing the impacts of an innovation, emphasizing the second point about geography and socioeconomic factors influencing the impact a given innovation has on a specific person or group of people.
- **Slide 13**: Some students have little/no understanding of how other people live in the world and the factors that limit impact or increase the divide among various groups of people based on geography, culture, and socioeconomic status. Have students discuss their thoughts on the factors in a person's life that would cause Napster's creation and use to have no impact. **Slide 14** gives specific reasons as to why a given person would be unaffected by a music-sharing (or any Internet-based) technology.
- **Slides 15-17**: Ask students if they've heard of the digital divide. Then present these slides, which define the term "digital divide" and discuss factors that contribute to it in the United States and on a global level. The slides contain several links and videos to supplement the information:
  - Table showing the estimated number of Internet users in each country in 2016: http://www.internetlivestats.com/internet-users-by-country/ (http://www.internetlivestats.com/internet-users-by-country/)
  - A PBS News Hour segment about the digital divide from 2013 (Show 0:00 – 3:27): https://www.youtube.com/watch?v=X537MiN6COI (https://www.youtube.com/watch?v=X537MiN6COI) – Video emphasizes that digital divide also includes digital literacy in addition to having access as well as how socioeconomic factors, age, race, etc. correspond to digital literacy and access.
  - Map showing percentage of Internet users by country in 2014: http://data.worldbank.org/indicator/IT.NET.USER.P2?view=map (http://data.worldbank.org/indicator/IT.NET.USER.P2?view=map) – Could ask students which regions of the world have the highest and lowest percentage of internet users per country.
  - An article from 2015 about technologies that Google and Facebook are separately developing in order to provide Internet access to places in the world that do not have the infrastructure – includes a short video (1:07): http://www.bbc.com/news/technology-34780127

    (http://www.bbc.com/news/technology-34780127)

Independent Analysis (20 min)

For this portion of the lesson, students will be analyzing the impacts of two innovations: artificial intelligence and the Find My Friends App. A worksheet for this activity ("Analyzing Impacts of Digital Innovations") can be found in the lesson resources folder (Lesson1_5wkst.docx).

- **Slide 18 (Innovation #1):** Students will be watching this video about artificial intelligence (0:00 – 5:12) https://www.youtube.com/watch?v=95KhuSbYJGE (https://www.youtube.com/watch?v=95KhuSbYJGE) . Point out the example on the worksheet table that has been given as a model of how to fill out the table. Give students time to fill out the front of the worksheet as they watch the video and a few minutes after watching the video to write down more of their thoughts. Then share out in groups and as a class. The specific contents of the video are as follows:
  - Basic assumption: It's unethical NOT to develop artificial intelligence.
  - Robots that learn, problem-solve, and are creative have been around since mid-20th century.
  - People are threatened: why hire people if you can have a robot?
  - Robots for everything: Google self-driving car, Watson going for diagnostic medicine, Perry Mastron for legal needs
  - Robots can do so much: not just boring, repetitive, dangerous, but also complex thinking jobs
  - Is it ethical to stop improvement? The ethic of truths: we need things like cheaper, better medical care.
  - The printing press challenged the status quo. Old inventions caused problems, too.
  - Ethics of progress: does the possibility of atrocity (nuclear bomb) mean we shouldn't develop nuclear power?
  - Is it unethical to stop the development of artificial intelligence?
  - (STOP at 5:12) after that it's about emotion. (skip it)
- **Slide 19 (Innovation #2):** Have students fill out the back of the worksheet as they read the article about the Find My Friends App: http://www.news.com.au/technology/gadgets/these-smartphone-users-share-how-tracking-app-find-my-friend-has-saved-them/story-fn6vihic-1226752063613 (http://www.news.com.au/technology/gadgets/these-smartphone-users-share-how-tracking-app-find-my-friend-has-saved-them/story-fn6vihic-1226752063613) (If doing this lesson without a computer, you'll need to print out copies of the article for students to read.) Allow students to discuss their findings with a partner or in groups. Get a few responses from groups if time permits.

Wrap Up (5 min)

- **Slide 20:** Students journal about the following prompt: If you come up with an innovation that solves a problem, what concerns do you need to consider before releasing it to the world? (Possible answers: Whom will it benefit or harm? Are there people it won't reach at all?)

Homework:

Students will analyze the innovation of 3D printing in the same manner they did for AI and the Find My Friends App, except they will be finding at least one online source on their own from which to draw their information. Provide students with the "Lesson 1.5 Homework" found in the lesson resource folder (Lesson1_5hw.docx) and provide them with the instructions given on **Slide 21.**

This assignment gives students practice analyzing the impacts of an innovation on their own, as well as attributing facts to a resource and the information to include for that resource for the Explore Performance Task. (The Explore PT is introduced later in the curriculum.) The worksheet Lesson1_5hw (in curriculum resources folder) can be used to support students, or they can write this information on a blank piece of paper, etc. The worksheet does not specify which innovation they are researching, so you could reuse it for future research related to impacts of innovations.

Options for Differentiated Instruction

For students who require more time for processing and writing down the impacts of AI, show the video a second time (the narrator in the video talks quickly).

If time is limited, split students up so that half of them are analyzing AI and the other half are analyzing the Find My Friends App. Have students review their findings with another student who analyzed the same innovation. Then have them jigsaw (https://www.teachervision.com/group-work/cooperative-learning/48532.html (https://www.teachervision.com/group-work/cooperative-learning/48532.html)) with students who analyzed the other innovation to share what they found.

Guided notes would be helpful for ELL or SpED.

Evidence of Learning

Formative Assessment

Journal:

- Have students respond to the following prompt:
  - If you come up with an innovation that solves a problem, what concerns do you need to consider before releasing it to the world?

Homework:

- Research the impacts of 3D printing.

Summative Assessment

Explore Performance Task

Internal Use Only

Pending Tasks

~~Need to move lesson folder (was 2-17)~~

~~Need to move this lesson earlier in unit 1~~

-

Future Work

Joe: ~~A future revison should further define how the teacher engages students with these topics, especially in the presentation, where the guidance to discuss the presentation may not be adequate.~~ Mostly redone D O-C

~~The ppt will still need to be redone because it directly uses published, commerical sources~~.

(http://csmatters.org) 1 - 6

0b1 - 0b110

A Problem Solving Process that Scales

Unit 1. Your Virtual World

Revision Date: Aug 21, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

Previous lessons in the "Your Virtual World" have investigated the impact of computer innovations on society. In this lesson, students will learn how using technology can enhance our abilities to solve larger and broader problems (problem solving). The lesson begins by examining reCAPTCHAs, which most students will be familiar with, but they may not realize how they solve two significant problems.

Outcomes

- Students will learn how computers are used to aggregate the computational power of individuals to solve large-scale problems through citizen science activities.
- Students will participate in a citizen science project. Aggregate problem-solving is sometimes called "crowdsourcing."

Overview

1. Getting Started (5 min) - Think-Pair-Share about reCAPTCHA.
2. Guided Activity (40 min) - Students examine citizen science and discuss its uses in the scientific community.
3. Wrap Up (5 min) - Journaling about potential additional uses of mass data collection.
4. Optional Activities.

Source

This lesson is an adaptation of a Code.org CS-P lesson from 2014.

Learning Objectives

CSP Objectives

- 1.2.1 - Create a computational artifact for creative expression.
  - 1.2.1C - Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming integrated development environments (IDEs), spreadsheets, three-dimensional (3-D) printers, or text editors.
  - 1.2.1D - A creatively developed computational artifact can be created by using nontraditional, nonprescribed computing techniques.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4B - Effective collaborative teams consider the use of online collaborative tools.
  - 1.2.4D - Effective collaboration strategies enhance performance.
- 7.1.2 - Explain how people participate in a problem solving process that scales.
  - 7.1.2A - Distributed solutions must scale to solve some problems.
  - 7.1.2B - Science has been impacted by using scale and "citizen science" to solve scientific problems using home computers in scientific research.
  - 7.1.2C - Human computation harnesses contributions from many humans to solve problems related to digital data and the Web.
  - 7.1.2D - Human capabilities are enhanced by digitally enabled collaboration.
  - 7.1.2E - Some online services use the contributions of many people to benefit both individuals and society.
  - 7.1.2F - Crowdsourcing offers new models for collaboration, such as connecting people with jobs and businesses with funding.
- 7.2.1 - Explain how computing has impacted innovations in other fields.
  - 7.2.1B - Scientific computing has enabled innovation in science and business.
  - 7.2.1E - Open and curated scientific databases have benefited scientific researchers.

7.1.2 ABCDEF

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.

Common Core ELA:

- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- WHST 12.1 - Write arguments on discipline specific content

NGSS Practices:

- 8. Obtaining, evaluation, and communicating information

Key Concepts

- Distributed computing solutions can be used to solve problems, collect data, assist with collaboration, and assist scientific research.

Essential Questions

- How does computing enhance human communication, interaction, and cognition?

Teacher Resources

Student computer usage for this lesson is: **required**

For the Student

- Access to the website Zooniverse:
  - https://www.zooniverse.org/#/projects (https://www.zooniverse.org/#/projects)
- Access to the article Crowdsourcing, For the Birds
  - http://www.nytimes.com/2013/08/20/science/earth/crowdsourcing-for-the-birds.html?pagewanted=all&_r=0 (http://www.nytimes.com/2013/08/20/science/earth/crowdsourcing-for-the-birds.html?pagewanted=all&_r=0)
- Access to the website Akinator.com or 20Q
  - http://en.akinator.com/ (http://en.akinator.com/)
  - http://www.20q.net/ (http://www.20q.net/)

For the Teacher

- Access the video Fight Spam and Save Shakespeare to show to the class
  - https://www.youtube.com/watch?v=VoybhowC4LE (https://www.youtube.com/watch?v=VoybhowC4LE)
- Additional Videos about Crowdsourcing:
  - https://www.youtube.com/watch?v=Buyub6vIG3Q (https://www.youtube.com/watch?v=Buyub6vIG3Q)
  - https://www.youtube.com/watch?v=-38uPkyH9vI (https://www.youtube.com/watch?v=-38uPkyH9vI)
- Crowdsourcing websites:
  - http://dailycrowdsource.com/training/crowdsourcing/what-is-crowdsourcing (http://dailycrowdsource.com/training/crowdsourcing/what-is-crowdsourcing)
  - http://education.nationalgeographic.com/education/idea/citizen-science-projects/ (http://education.nationalgeographic.com/education/idea/citizen-science-projects/)
  - https://www.zooniverse.org/ (https://www.zooniverse.org/)

Lesson Plan

Getting Started (5 min)

Think-Pair-Share: Solving problems with reCAPTCHA

1. Show an image of a reCAPTCHA (by visiting a website that employs this technology or from an image search).
2. Direct students to describe for their elbow partner when and where they have encountered this on the web and discuss why it is used.
3. As a class, view the video Fight Spam and Save Shakespeare and discuss:
   - What two problems are being solved with reCAPTCHA?
   - How has reCaptcha used the aggregate computing power of millions of people to solve problems in digitizing old books?
   - Have you heard of other aggregate solutions to problems that are currently being solved?
   - Are there other problems you can think of that a strategy of capturing the work of millions of people and their computers might be able to solve?

Guided Activities (40 min)

Part 1 (10 min) - Learn about tracking birds with citizen science

- As a class, read the article Crowdsourcing, For the Birds
- *Optional website and videos on crowdsourcing*:
  - Daily Crowdsourcing, "What is Crowdsourcing?" http://dailycrowdsource.com/training/crowdsourcing/what-is-crowdsourcing (http://dailycrowdsource.com/training/crowdsourcing/what-is-crowdsourcing)
- *Videos*:
  - What is crowdsourcing? (2:50) https://www.youtube.com/watch?v=Buyub6vIG3Q (https://www.youtube.com/watch?v=Buyub6vIG3Q)
  - Crowdsourcing and Crowdfunding Explained (3:48) https://www.youtube.com/watch?v=-38uPkyH9vI (https://www.youtube.com/watch?v=-38uPkyH9vI)

Part 2 (15 min) - Participate in citizen science

- Direct students to the Zooniverse website: https://www.zooniverse.org/#/projects (https://www.zooniverse.org/#/projects)
- Allow students to participate several times. Encourage them to pick a different project each time.
- As a class, briefly discuss how the student responses will help scientists.
- Discuss with the class the **components** of citizen science.
  - **A problem to be solved.** (How can a computer be used to identify objects? How can a program learn about those objects?)
  - **A way for people to participate.** (Create data for identifying objects, ex: Plankton)
  - **A website or app to aggregate data.** (The Zooniverse website)
  - **A way to turn the data into knowledge.** (Students can brainstorm possible algorithms to handle the data captured.)
- Ask students if they know of other citizen science projects (an Internet search will turn up dozens, possibly some in your area).
- *Optional*: have students work for 5 minutes in a group to identify the **components** of a selected citizen science project. Have each group share their ideas with the class.
- Engage students in a discussion of problems that they think a citizen science project could address using computers to harness the power of data from many individuals.

Part 3 (15 min) - Guessing what you are thinking

Play an on-line game which aggregates human information. (Direct students to Akinator.com or 20Q. If these website do not work on student computer, teacher can display the website and students participate as a whole class.)

- Discuss how the game acts intelligently.
- Have students work in small groups to devise a method to collect data to teach a computer how to play 20 questions. Each group should share their ideas.

Wrap Up (5 min)

In their journals, have students describe a *mobile app* that uses multiple user input to collect data. Emphasize data collection that would be beneficial to high school students.

Optional Additional Activities

Add knowledge to Wikipedia

The purpose of this activity is for students to contribute their knowledge to the aggregated collection of knowledge known as "Wikipedia."

- Divide the class into teams of two.
- Instruct students to think of topics they have some knowledge of: a sport, music, their community, a hobby, or other expertise. Direct them to explore the content on that topic on Wikipedia.
- Each team should work independently to learn how to modify a Wikipedia page and then add some content to the topic. Encourage students to be independent learners by working to learn how to make edits with as little help as possible from the teacher.
- At the end of class summarize student learning through a discussion or by summarizing the steps of the Wikipedia editing process in their journals. Additionally, students could collaboratively write a paragraph about how they worked together to complete the task.

Learn about Kickstarter

- Direct students to research Kickstarter.com. Ask them to choose a current project that they think is worthy of funding and justify that judgment. Suggest that the students think of a project that they would like to see as a good candidate for Kickstarter.

Learn about Waze

- Have students read and discuss this article http://allthingsd.com/20130719/after-waze-what-else-can-mobile-crowdsourcing-do/ (http://allthingsd.com/20130719/after-waze-what-else-can-mobile-crowdsourcing-do/).

Use Search Trends as Predictors

- Ask students to learn about how search trends can be used as predictors. They should share what they learned with the class by any creative means.

More Citizen Science

- Find a citizen science project that is of interest to you. Participate in the project. Report your experiences in any creative format: a report, a diorama, a website, video, etc.

Picture Stitching

"Picture Stitching" is the practice of blending hundreds of photos to create one huge detailed picture.

- Investigate the stitched photo of the 365-gigapixel image of Mont Blanc that was created by stitching together 70,000 images http://www.in2white.com/# (http://www.in2white.com/) .

Evidence of Learning

Formative Assessment

During the activity, can students identify crowdsourcing or a citizen science project?

Summative Assessment

Sample assessment questions:

- Explain the dual purposes of a reCAPTCHA.
- Explain how people can add value to citizen science projects, using several examples.
- Explain how people can add value to an on-line guessing game.
- Create a possible flow chart or description of how data collected online can be used to help computers learn.

Internal Use Only

Future Work

I think these lessons may need modification based on the experience of the master teacher cohort, in terms of being more directive with students about what they should be learning. -mdj

(http://csmatters.org) 1 - 7

0b1 - 0b111

Unit 1 Assessment

Unit 1. Your Virtual World

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary:

This is the unit assessment for the first unit of the AP Computer Science Principles curriculum - Your Virtual World. This curriculum provides both a testbank of questions with answers for objective questions, and a sample final unit exam prototype for teacher use extracted from the testbank.

Outcomes:

- Students will have demonstrated their learninging in this unit through answers to both objective and essay questions

Overview:

1. Hand out assessment (2 min)
2. Students write answers to assessment (46 min)
3. Collect and grade the assessment (2 min)

Learning Objectives

Common Core ELA:

- RST 12.2 - Determine central ideas and conclusions in the text
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.7 - Integrate and evaluate multiple sources of information presented in diverse formats and media
- RST 12.8 - Evaluate the hypotheses, data, analysis, and conclusions in a science or technical text
- RST 12.9 - Synthesize information from a range of sources
- RST 12.10 - Read and comprehend science/technical texts
- WHST 12.1 - Write arguments on discipline specific content
- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.4 - Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience
- WHST 12.5 - Develop and strengthen writing as needed by planning, revising, editing, rewriting
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products

NGSS Practices:

- 3. Planning and carrying out investigations
- 4. Analyzing and interpreting data
- 5. Using mathematics and computational thinking
- 7. Engaging in argument from evidence
- 8. Obtaining, evaluation, and communicating information

NGSS Content:

- HS-ETS1-1. Analyze a major global challenge to specify qualitative and quantitative criteria and constraints for solutions that account for societal needs and wants.

Key Concepts

This assessment ascertains that students have a basic understandings of all the concepts presented in the unit, therefore, all learning objectives are assessed in this unit.

Essential Questions

- How can a creative development process affect the creation of computational artifacts?
- How can computing and the use of computational tools foster creative expression?
- How can computing extend traditional forms of human expression and experience?
- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How does computing enhance human communication, interaction, and cognition?
- How does computing enable innovation?
- What are some potential beneficial and harmful effects of computing?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

Students should be aware of the magnitude of impact on individuals and society that result from technological advancements in computing, as well as the rapid pace of change that occurs because of new developments.

Teacher Resources

Student computer usage for this lesson is: **optional**

In the Lesson Resources folder:

- "Unit 1 Summative Assessment Test DATABASE - AP CSP"
  - the compiled Unit Test testbank
- "Unit 1 Summative Sample Test student copy- AP CSP"
  - a sample exam

- "Unit 1 Summative Assessment DATABASE ANSWERS"
  - answer key for the Unit Test testbank
- "Unit 1 Summative Sample Test ANSWERS- AP CSP"
  - answer key for sample exam
- The instructor will have to review and develop their own evaluation of the essay questions.

Lesson Plan

Getting Started (2 min)

Make sure each student has a copy of the assessment and the necessary writing instruments.

Independent Activity (46 min)

Allow one 45-50 minute class session to administer this assessment.

Distribute the Unit Assessment Test, which consists of 25 objective questions and a choice of four essay prompts. (Teachers can instruct students in selecting one or more short essay responses for students to answer, based on teacher preferences and time allotments.)

Wrap Up (2 min)

Collect all papers from the students.

Evidence of Learning

Summative Assessment

This is the summative unit assessment for Unit 1 - Your Virtual World. A sample summative test and a testing databank of questions are provided by the curriculum. (Note that the sample assessment may not be appropriate for some classes, depending on the particular focus that the teacher has taken -- e.g., the sample assessment includes hexadecimal conversions, which are an optional component of Lesson 1-4.)

Internal Use Only

Future Work

It would be good to develop a rubric for grading the essay questions that is consistent with the writing required for the performance task.

(http://csmatters.org) 1 - 8

0b1 - 0b1000

Optional

The Basics of Research and Technical Writing

Unit 1. Your Virtual World

Revision Date: Oct 09, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Pre-lesson preparation

Review the Teaching Technical Writing slides to help prepare teaching the topics of research and writing. It may be wise to review citation styles and pick one you want your students to use -- as long as they are consistent, the particular style should not matter. For Session 2, you can print out the Cut it Out activities from the slides for your students if you want them to try for themselves on paper.

Summary

Students will learn the basics of technical writing and research, practicing skills including finding good sources, citing properly, and differentiating between quoting, summarizing, and plagiarism.

Outcomes

- Students will practice the style and process of technical writing.
- Students will learn and employ practical writing advice.
- Students will identify good and bad sources of information.
- Students will learn the importance of citing sources.
- Students will recognize there are different styles of citations.
- Students will quote and summarize from information sources.
- Students will understand the definition and consequences of plagiarism.
- Students will differentiate between plagiarism and proper attribution.

Overview

Session 1

1. Getting Started (5 min)
    - What is research?
2. Activity (15 min)
    - Finding good sources
3. Activty (25 min)
    - Plagiarism vs. Quotation vs. Paraphrasing
4. Wrap-up (5 min)

Session 2

1. Getting Started (5 min)
    - Who writes?
2. Guided Activity (40 min)
    - Cutting out useless words
    - Practice assessing written responses
3. Wrap-up (5 min)

Learning Objectives

CSP Objectives

- 7.2.1 - Explain how computing has impacted innovations in other fields.
    - 7.2.1B - Scientific computing has enabled innovation in science and business.
    - 7.2.1C - Computing enables innovation by providing the ability to access and share information.
- 7.5.1 - Access, manage, and attribute information using effective strategies.
    - 7.5.1A - Online databases and libraries catalog and house secondary and some primary sources.
    - 7.5.1B - Advance search tools, Boolean logic, and key words can refine the search focus and/or limit search results based on a variety of factors (e.g., data, peer-review status, type of publication).
    - 7.5.1C - Plagiarism is a serious offense that occurs when a person presents another's ideas or words as his or her own. Plagiarism may be avoided by accurately acknowledging sources.
- 7.5.2 - Evaluate online and print sources for appropriateness and credibility.
    - 7.5.2A - Determining the credibility of a source requires considering and evaluating the reputation and credentials of the author(s), publisher(s), site owner(s), and/or sponsor(s).
    - 7.5.2B - Information from a source is considered relevant when it supports an appropriate claim or the purpose of the investigation.

Common Core ELA:

- RST 12.1 - Cite specific textual evidence to support analysis of science and technical texts, attending to important distinctions the author makes and to any gaps or inconsistencies in the account.
- RST 12.2 - Determine central ideas and conclusions in the text
- RST 12.5 - Analyze how the text structures information or ideas into categories or hierarchies
- RST 12.6 - Analyze the author's purpose in providing an explanation, describing a procedure
- RST 12.7 - Integrate and evaluate multiple sources of information presented in diverse formats and media
- RST 12.8 - Evaluate the hypotheses, data, analysis, and conclusions in a science or technical text

- RST 12.9 - Synthesize information from a range of sources
- RST 12.10 - Read and comprehend science/technical texts
- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.4 - Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience
- WHST 12.5 - Develop and strengthen writing as needed by planning, revising, editing, rewriting
- WHST 12.7 - Conduct short as well as more sustained research projects to answer a question
- WHST 12.8 - Gather relevant information from multiple authoritative print and digital sources, using advanced searches effectively; assess the strengths and limitations of each source
- WHST 12.9 - Draw evidence from informational texts to support analysis, reflection, and research

NGSS Practices:

- 1. Asking questions (for science) and defining problems (for engineering)
- 7. Engaging in argument from evidence
- 8. Obtaining, evaluation, and communicating information

Key Concepts

- Research involves finding good sources of information and synthesizing knowledge to convey your understanding to others.
- Attribution of ideas, information, and quotes is an important facet of research.
- Plagiarism is a complex unethical practice that is not limited to simply copying language, and softer forms of plagiarism should be understood and avoided.
- Differentiating the quality of one source from another can be challenging, and often relies on contextual information (such as author, location, date of publication).

Essential Questions

- How does computing enhance human communication, interaction, and cognition?

Teacher Resources

Student computer usage for this lesson is: **required**

Additional resources on the basics of research include the following. Keep in mind that what will benefit students the most for this lesson is to focus on tips and guidelines related to content rather than generic advice like avoiding too many adverbs.

- EasyBib's Research Quick Guide:
  - http://www.easybib.com/guides/students/research-guide/research-quick-guide/ (http://www.easybib.com/guides/students/research-guide/research-quick-guide/)
- EasyBib's Plagiarism Flowchart:
  - http://www.easybib.com/guides/students/research-guide/what-is-plagiarism/ (http://www.easybib.com/guides/students/research-guide/what-is-plagiarism/)
- College Board has a separate AP course devoted to research (and related, general skills). They provide an outline of that course in the AP Research Course Framework (includes Learning Objectives and Essential Knowledge related to research skills):
  - https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-research-course-and-exam-description.pdf (https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-research-course-and-exam-description.pdf)
- Sharpen your academic skills (video on "Getting Started on Research")
  - http://stemtransfer.org/resources/posttransfer/academic-skills/ (http://stemtransfer.org/resources/posttransfer/academic-skills/)

Lesson Plan

Session 1

Getting Started (5 min)

Discussion: What is research?

Refer to the "Teaching Technical Writing" slides in the Resources folder for an overview on technical writing and the process of research as well as advice on teaching those topics.

- To broach the topic of research and provide some basic definitions, start off with a guided discussion and pose the following questions to your students:
  - What kinds of research have you done before? What exactly did you do? Did you follow a process? What was the hardest part?

    - What is research? What are the steps of research?
    - Writing a research report is sometimes referred to as technical writing. How does technical writing differ from other kinds of writing, like the kind you might do in an English class?
- Remind students about the upcoming performance task and the expectations.
- The first step of research is picking a topic and finding sources, which segues into the next activity. For the purposes of practice, today's research topic is already chosen.
- A key insight to grasp is the difference between researching for yourself alone and researching to later convey your understanding through writing. When searching for information on your own, you may be satisfied with simpler, summarizing sources such as Wikipedia. However, when you intend to communicate and transmit research to others, you become responsible for its quality. For that reason both the quality of your original sources and attributing them (that is, not plagiarizing) become of paramount importance.

Activity (15 min)

Discussion about good sources of information and what meets that criteria [5 min]

- As a class, have your students try to list as many sources of information they can think of (up to a reasonable amount). Then ask which of these would be considered "good" sources of information, and why? What criteria make one source better than another? Can they think of a "bad" source of information? What makes it that way?

Finding good sources of information [10 min]

- Each student receives a copy (paper or electronic) of the "Website Evaluation Guide"
- Several things to note and keep in mind while using the EasyBib guide:
    - The criteria for evaluating websites are at **the end** of the guide. Some of the guidelines are vague; you may want to discuss a few precise examples with students.
    - When doing research, the most crucial criteria for evaluating any source are the author's purpose and goal behind writing their article. Why did they write what they wrote? Students should always question an author's motivations while assessing sources (not just as a challenge, but to really understand the author and greater context). Is there some monetary reason for their authorship (which is not necessarily bad) such as they are journalists and it is their job, or they are running a website to support advertisements? Does the author have an immediate or indirect connection to the topic they are covering, such as are they an expert on the topic or an outside observer? Does the author have any potential political or other partisan reason for writing their source?
    - The lack of a named author or editor is not automatically a mark against a source, depending on the context. Some sources publish good information without a specific author, such as Associated Press articles that lack bylines.
    - In terms of authorship, the EasyBib guide uses examples such as "the author is a journalist" as a positive source in contrast with someone with "journalistic experience" as a negative source. These distinctions are sometimes unclear. While verifiable credentials may help assess trustworthiness of an author, some credentials may be fabricated or otherwise made to appear legitimate. Even verifiable journalists and other authors may be untrustworthy on certain topics or under certain circumstances. Again, the greater context matters. Students should be aware of this possibility and always ask themselves what an author's motivations for writing the given article may be.
    - A website or other source adopting legitimate-sounding names or titles (such as "Encyclopedia") should not factor into positive criteria. Some sources will mislead for advertising or other purposes by taking lengths to appear credible.
    - In terms of "currency" and up-to-dateness, both the context and purpose matter. A news source with an article that is no longer current but was covering news at the time is not necessarily a bad source. Similarly, informative articles that are several years old can still be valid depending on their purposed and topics they cover.
- Have students review the "Finding sources for your research" handout.
- A difference that students should keep in mind is the one between content distribution sources and content producers. In the case of a service like YouTube, the site itself is not a source (unless it is an official video from the company) -- the user who originally produces the videos is the source. So the fact that a video on YouTube does not inherently make it good or bad; it depends on the legitimacy of the video maker.

Activity (25 min)

Discussion about quoting, paraphrasing with attribution, and plagiarism [15 min]

- Use the "Writing Tips: Plagiarism vs. Citation" slides in the Resources folder to aid the discussion and give examples of plagiarism
- All academic writing requires appropriate attribution. The purpose of citation is multifaceted: it credits the original thinker, it establishes a chain of evolving ideas, and it demonstrates the writer has done a suitable job researching sources. Citing sources properly lends credibility to the researcher and is a necessary persuasive tool.
- Show or discuss an example citation (ideas: a quote, an in-text parenthetical citation, a citation from a paper's bibliography)
- Rule of thumb: if you are not the original source of an idea or fact that is not common knowledge, you should cite it!
- Any sentence you write that relies on an idea that is not your own should be cited. If the whole paragraph is about an idea from the same source, you can place the citation at the end of the paragraph.
- Discuss the definitions and difference among quoting, paraphrasing, and plagiarizing a source.
    - Plagiarism is taking credit for someone else's ideas as if they were your own.
        - Plagiarism is dishonest and unethical, and it ultimately undermines the credibility of the plagiarist.
        - The consequences of plagiarism are quite severe: you can be expelled from college, be fired from your job, and even have your degree revoked if you plagiarize.
    - Quotation is taking words verbatim from a source, surrounding those words with quotation marks, and specifically naming who said the quote and when/where.
        - In general, quotes should be no longer than a sentence or two at most.

        - Ideally, quotations should only be used as secondary evidence or to illustrate a point already made in your writing.
        - Quotation is not plagiarism as long as it is properly attributed.
        - Improper quotation (lifting text verbatim without using double quotes and/or stating the original source) is plagiarism.
    - Paraphrasing is taking words or ideas from someone and rewriting or condensing them in your own writing.
        - Try to be as clear as possible when you are paraphrasing to convey the source and what it is you are paraphrasing
        - Paraphrasing can be plagiarism:
            - Near-verbatim paraphrasing (just changing or omitting a few words or swapping in synonyms) is plagiarism
            - Summarizing without citing the source is plagiarism
- Finish with the examples in the slides

Practice judging plagiarism vs. appropriate citations [10 min]

- Give your students the handouts for "Examples of Paraphrasing" and "Plagiarism vs. Paraphrasing Exercise" (in the Resources folder) and have them complete it in pairs, groups, or as a class. Go over the correct answers and have them discuss why they labelled each.

Wrap-up (5 min)

Journal Entry

With the time remaining, have students reflect and write in their journal what topics they are considering for their performance task and which sources of information they plan to find first.

Session 2

Getting Started (5 min)

Discussion: Who writes?

Use the "Writing Tips Process and Style" slides in the Resources folder to guide this lesson and begin with the questions and overview on slides 2 and 3.

Guided Activity (40 min)

Interactive Lesson: The Writing Process [20 min]

- Have your students take notes. Warn them that this will be used in their homework!
- Continuing with the slides, go over the Writing Process and Style sections
- For the "Cut it Out" exercises in the slides, you may want to have your students follow along on print-outs of the slides.
- Continue through the rest of the slides

Group Activity [20 min]

- Pass the "Assessing Sample Performance Task Responses" handouts to your students.
- In pairs or groups, have your students complete the first handout, and before moving onto the next one, discuss with a group your decisions on how to assess the writing (the Assessments folder should have an example key)
- Complete the remaining handouts in the same fashion

Wrap-up (5 min)

Journal Entry or Homework: two options

Have your students reflect and write responses to one or both of the following:

- Write in your journal how you think technology and computers have changed research and writing over the past hundred years.
- Make a bulleted list of 10 writing tips you were taught about today (which you can reuse when writing your performance tasks in the future).

Evidence of Learning

Formative Assessment

Session 1 Journal:

- Have your students reflect and write in their journal what topics they are considering for their performance task and which sources of information they plan to find first.

Session 2 Journal / Homework:

- Have your students reflect and write responses to one or both of the following:
    - Write in your journal how you think technology and computers have changed research and writing over the past hundred years.

- Make a bulleted list of 10 writing tips you were taught about today (which you can reuse when writing your performance tasks in the future).

(http://csmatters.org) 1 - 9

0b1 - 0b1001

Practice Performance Task

Unit 1. Your Virtual World

Revision Date: Sep 10, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary: Students will complete a practice partial Explore Performance Task by creating an artifact of the student's choosing. This is the student's first exposure to a CS Principles Performance Task; therefore, this is a guided lesson.

Outcomes:

- Students will use the Internet to research a technology of the student's choice and create an artifact to report on the research.

Overview:

Session 1:

1. Getting Started (5 min)
    1. A quick activity that requires an effective internet search
2. Activity (40 min)
    1. Critical Reading Skills Activity [15 min]
    2. Brainstorming Activity [10 min]
    3. Practice Task Activity [15 min]
3. Wrap-up (5 min)

Session 2:

1. Getting Started (5 min)
2. Guided Activity (40 min)
    - Work on practice performance task
3. Wrap-up (5 min)

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
    - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
    - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.1 - Create a computational artifact for creative expression.
    - 1.2.1A - A computational artifact is something created by a human using a computer and can be, but is not limited to, a program, an image, an audio, a video, a presentation, or a Web page file.
    - 1.2.1B - Creating computational artifacts requires understanding of and use of software tools and services.
    - 1.2.1C - Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming integrated development environments (IDEs), spreadsheets, three-dimensional (3-D) printers, or text editors.
    - 1.2.1E - Creative expressions in a computational artifact can reflect personal expressions of ideas or interests.
- 1.3.1 - Use computing tools and techniques for creative expression.

- 7.2.1 - Explain how computing has impacted innovations in other fields.
  - 7.2.1C - Computing enables innovation by providing the ability to access and share information.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
  - 7.3.1A - Innovations enabled by computing raise legal and ethical concerns.
- 7.4.1 - Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts.
  - 7.4.1A - The innovation and impact of social media and online access varies in different countries and in different socioeconomic groups.
  - 7.4.1B - Mobile, wireless, and networked computing have an impact on innovation throughout the world.
- 7.5.1 - Access, manage, and attribute information using effective strategies.
  - 7.5.1A - Online databases and libraries catalog and house secondary and some primary sources.
  - 7.5.1C - Plagiarism is a serious offense that occurs when a person presents another's ideas or words as his or her own. Plagiarism may be avoided by accurately acknowledging sources.
- 7.5.2 - Evaluate online and print sources for appropriateness and credibility.
  - 7.5.2A - Determining the credibility of a source requires considering and evaluating the reputation and credentials of the author(s), publisher(s), site owner(s), and/or sponsor(s).

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.7 - Integrate and evaluate multiple sources of information presented in diverse formats and media
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products
- WHST 12.7 - Conduct short as well as more sustained research projects to answer a question

NGSS Practices:

- 3. Planning and carrying out investigations

Key Concepts

- Students will revisit concepts learned in Unit 1 to complete a teacher-guided mini-performance task.
- The teacher will utilize different reading and writing strategies to assist the students in interpreting a performance task and creating an artifact.
- Students may need guidance in selecting a format for their artifact. Students should be encouraged to create an artifact using a tool they are already familiar with so the lesson can focus more on research, applying Unit 1 lessons and creativity.

Essential Questions

- How can a creative development process affect the creation of computational artifacts?
- How can computing and the use of computational tools foster creative expression?
- How can computing extend traditional forms of human expression and experience?
- How does computing enhance human communication, interaction, and cognition?
- How does computing enable innovation?
- What are some potential beneficial and harmful effects of computing?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- Presentation with sample visual artifacts: artifact.pptx
- Practice Performance Task Rubrics:
  - CS Matters version (uses 5-point scale): Unit 1 Practice Performance Task Rubric CSM.docx
  - College Board version (uses 3-point scale): Unit 1 Practice Performance Task Rubric CB.docx
- Practice Performance Task:
  - College Board version for both Investigate (no longer part of CS Principles), Explore (visual and written artifacts), and Create tasks (for teacher reference only): CS_Principles_Performance_Assessment2014-1-9.pdf
  - Explore task only (useful for sharing with students): Explore Performance Task.pdf

For the Teacher:

- Pre-reading strategies: http://www.studygs.net/preread.htm (http://www.studygs.net/preread.htm)

Lesson Plan

Session 1

Getting Started (5 min)

A quick activity that requires an effective Internet search

- Example: Count your clicks! In as few “clicks” as possible, locate a web site for a physical store within 50 miles of your school where you can purchase a blue laptop bag.

Activity (25 min)

Explore Performance Task - Introduction

- Each student receives a copy (paper or electronic) of the Explore performance task (Explore Performance Task.pdf) and the CS Matters scoring rubric (Unit 1 Practice Performance Task Rubric CSM.docx). ***Note:*** *This is the official College Board version, which has not been updated fully for the new version of the rubrics. In particular, the rubric now restricts the permitted file types, and the resource dates in the performance task description are outdated. Those rubric requirements are more current and should be followed by the students. Note that the rubric is still likely to change in the future; we have developed a more fine-grained scoring rubric that we recommend using, to give the students more feedback and guidance. (The College Board rubric uses a 3-point scale; the CS Matters rubric is on a 5-point scale.)*
- Use the provided presentation (artifact.pptx) to give an overview of the requirements for the visual component of the Explore task, and to show the students several example artifacts. Lead the students in a discussion of how the artifacts do (or do not) meet the requirements in the rubric and task description.
- Use a variety of pre-reading activities to assist students' understanding of the requirements of the task:
  - Read the section headers.
  - Make predictions of what the students will be doing for the performance task.
  - Notice the structure of the text: introduction and short description, followed by details of the requirements.
  - Make connections between what was learned in Unit 1 with the requirements of the task.
- Students should read the performance task carefully. Encourage highlighting key terms. If necessary, work with students to transfer the requirements from paragraph form to a bulleted list.

Activity (5 min)

Brainstorming

1. Students should refer to the topic they chose at the end of Lesson 3. They may either choose this topic to research, or pick another topic from the following choices:
   - Social media
   - Online retail and banking and online businesses
   - Cloud data storage
   - Government surveillance
2. If they did not do so at the end of Lesson 3, each student should pick an innovation that connects to the topic chosen. They will research this innovation to create their practice artifact.

(Limiting the choices will allow students to compare and contrast their final artifacts, facilitate self-assessment, and make it possible to identify exemplars of each option.)

Activity (10 min)

Practice Performance Task

Students should begin the practice performance task. (This work will continue in the next class session.) During the class, the teacher should provide guidance to students to stay focused on the outcomes of the task, assist students in breaking down the task into workable steps, and check for students' understanding of the task.

Wrap-up (5 min)

Journal Entry

Give students a few minutes warning before the end of class. With 4 or 5 minutes remaining, have students reflect and write in their journal a verbal snapshot of the artifact they are creating.

Session 2

Getting Started (5 min)

- Have students write a goal for the class period in their journal. "During this class, I will complete: ..."
- Spend 2 or 3 minutes discussing difficulties encountered the previous day and possible solutions.

Activity (40 min)

Practice performance task

- Students continue work on the practice task. Continue providing guidance.
- *Note*: This practice task may need more than two class periods to complete. It may be necessary to break the task into chunks and assign some portions for homework. It would also be beneficial to spend class time after completion to have students present their artifacts to the class, identify exemplars, and discuss lessons learned.

Wrap-Up (5 min)

Options for Differentiated Instruction

Some students will benefit from having a teacher-selected topic and a step-by-step plan for completing the task.

Evidence of Learning

Formative Assessment

While the students are completing the performance task, check for understanding by asking the students:

- What technology are you researching?
- What artifact will you be creating? What is your plan?

Summative Assessment

The artifact from this Practice Performance task can be used as a summative assessment using the rubric in the Teacher Resources

Internal Use Only

Future Work

This lesson should not be based on a single practice performance task. A variety of practice tasks should be generated and available for teachers' use.

(I don't know what the above comment refers to - who wrote it? -mdj)

(http://csmatters.org) 2 - 1

0b10 - 0b1

Programming: Introduction and Motivation

Unit 2. Developing Programming Tools

Revision Date: Oct 01, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

Students will be introduced to programming for the first time. They will learn about computer science, computing for good, some of the potential outcomes of programming, and the definition of abstraction.

Outcomes

- Students will learn various creative and helpful purposes for programming.
- Students will learn about abstraction.
- Students will create Python code in Runestone.

Overview

1. Getting Started (10 min)
2. Guided Activities (40 min)
    1. Activity 1: ThinkPair-Share/Write-Pair-Share [5 min]
    2. Activity 2: Computing for Good [10 min]
    3. Activity 3: Do the 5 sections of the General Introduction in Runestone. [25 min]
3. Optional homework: Read and write a tweet about one of the Seven things you should know if you're starting out programming [10 min]

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
- 1.2.1 - Create a computational artifact for creative expression.
    - 1.2.1B - Creating computational artifacts requires understanding of and use of software tools and services.
    - 1.2.1C - Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming integrated development environments (IDEs), spreadsheets, three-dimensional (3-D) printers, or text editors.
    - 1.2.1D - A creatively developed computational artifact can be created by using nontraditional, nonprescribed computing techniques.
    - 1.2.1E - Creative expressions in a computational artifact can reflect personal expressions of ideas or interests.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
    - 1.2.2B - A creative development process for creating computational artifacts can be used to solve problems when traditional or prescribed computing techniques are not effective.
- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
    - 1.2.3C - Combining or modifying existing artifacts can show personal expression of ideas.
- 3.1.2 - Collaborate when processing information to gain insight and knowledge.
    - 3.1.2A - Collaboration is an important part of solving datadriven problems.
    - 3.1.2B - Collaboration facilitates solving computational problems by applying multiple perspectives, experiences, and skill sets.
    - 3.1.2C - Communication between participants working on data-driven problems gives rise to enhanced insights and knowledge.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
    - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
    - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
- 5.2.1 - Explain how programs implement algorithms.
    - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
    - 5.2.1B - Program instructions are executed sequentially.
- 5.3.1 - Use abstraction to manage complexity in programs.
    - 5.3.1H - Data abstraction provides a means of separating behavior from implementation.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
    - 7.1.1E - Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results.
    - 7.1.1M - The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.
    - 7.1.1N - The Internet and the Web have changed many areas, including e-commerce, health care, access to information and entertainment, and online learning.
    - 7.1.1O - The Internet and the Web have impacted productivity, positively and negatively, in many areas.
- 7.4.1 - Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts.
    - 7.4.1A - The innovation and impact of social media and online access varies in different countries and in different socioeconomic groups.
    - 7.4.1B - Mobile, wireless, and networked computing have an impact on innovation throughout the world.
    - 7.4.1C - The global distribution of computing resources raises issues of equity, access, and power.
    - 7.4.1D - Groups and individuals are affected by the "digital divide" - differing access to computing and the Internet based on socioeconomic or geographic characteristics.
    - 7.4.1E - Networks and infrastructure are supported by both commercial and governmental initiatives.

Common Core ELA:

- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases

- RST 12.10 - Read and comprehend science/technical texts
- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.9 - Draw evidence from informational texts to support analysis, reflection, and research

Key Concepts

Students must understand that stereotypes of computer programmers are not accurate, and that 'coding' is something everyone can learn to do. Computing can be a creative expression and used for good.

Possible misunderstandings: The term NGO is used in the article *Programming for Good: The Story of Code for India*, without explicitly defining it as Non-Governmental Organization.

Abstraction is a tricky idea. Python allows us to describe what we want to do such as "print" and "input" because it provides the details that explain to the computer how to accomplish the task of taking many keypresses followed by a press of the Enter key to allow the computer to store the information we entered, and also knows how to take information stored in the computer's memory and cause it to appear as a series of recognizable dots on the screen using print.

Essential Questions

- How can computing and the use of computational tools foster creative expression?
- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How can computation be employed to help people process data and information to gain insight and knowledge?
- What opportunities do large data sets provide for solving problems and creating knowledge?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How does abstraction make the development of computer programs possible?
- How does computing enhance human communication, interaction, and cognition?
- How does computing enable innovation?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

Teacher Resources

Student computer usage for this lesson is: **required**

TEACHERS need to have the class and user accounts set up in Runestone to track student progress.

- **http://interactivepython.org/runestone/static/thinkcspy/toc.html (http://interactivepython.org/runestone/static/thinkcspy/toc.html) Create your own classroom**

For the Students:

- *Programming allows us to solve problems and use computing for good* articles used in this lesson:
  - http://www.attendly.com/programming-for-good-the-story-of-code-for-india (http://www.attendly.com/programming-for-good-the-story-of-code-for-india/)
- *Seven things you should know when you're starting out programming* article:
  - http://www.theguardian.com/info/developer-blog/2011/oct/07/programming-developer-journalist (http://www.theguardian.com/info/developer-blog/2011/oct/07/programming-developer-journalist)
- "Meet a computer scientist" to see the diveristy in computer science
  - http://mcwic.github.io/htmlblocks/computerscientistlibrary.html (http://mcwic.github.io/htmlblocks/computerscientistlibrary.html)
- Creating a Student Account in Runestone (Make sure to add your Runestone class number before printing)
  - https://docs.google.com/document/d/1FlyM5-jHIcDWCTOCnInS3ZDNinAECRybqoCqTc03VzQ/edit?usp=sharing (https://docs.google.com/document/d/1FlyM5-jHIcDWCTOCnInS3ZDNinAECRybqoCqTc03VzQ/edit?usp=sharing)

Note: If computer is not being used, students will need their own copies of the articles

Lesson Plan

Getting Started (5 min)

[use the Presentation on Programming Python in Runestone]

Journal Entry: What are some ways that writing programs is a creative endeavor? [slide 2]

Share answers with your elbow partner. Then share answers with class

Note to the teacher: This lesson comes immediately after the practice Explore performance task. To provide more time for sharing and discussion, you may want to assign the reading and questions from Activity 1 as homework in addition or instead of the optional homework.

Guided Activities (40 min)

Activity 1: Computing for Good [15 min]

Open Discussion: What are some ways you know that computing has been used for "good?" [slide 3]

Go to *Programming for Good: The Story of Code for India*

http://www.attendly.com/programming-for-good-the-story-of-code-for-india/ (http://www.attendly.com/programming-for-good-the-story-of-code-for-india/)

Read articles or selected text in pairs, with alternating pairs each reading one article. Pairs of pairs get together to share what they read and what they got out of the article.

In pairs, answer the following questions:

1. How did the impact of Hurricane Sandy in 2012 begin the concept of Code for India?
2. What is the reason for the Adopt-a-School app, and what is the result of its use?
3. What is the reason for the Spotter app?
4. What is the reason for the Bravehearts app, and what is its significance for public safety? Where else would this app be useful?
5. If you were able to design an app "for good," what problem would you try to solve?

Check for Understanding: Teams should share their answers to their instructor.

Activity 2: Coding with Python in Runestone [25 min]

[use the Presentation on Programming Python in Runestone slides 4-end]

1. Runestone was used in Unit 1 Lesson 6. [slide 4]
   Use that link and account information to access the eTextbook. BE SURE YOU HAVE SET UP STUDENT ACCOUNTS
   Read The Way of the Program and answer questions on the student handout.
2. Go to the next section: Algorithms and check to see that all student answers are being correctly recorded in Runestone. Display answer results on the teacher screen. Have students continue to fill in answers on student handout. [slide 5]
3. Have students independently complete the next 3 sections (or finish them for homework) and fill in the student handout.
4. Skip ahead to A Typical First program [slides 7-13]
   1. Demonstrate how to run the program, change the code, add an input line, and display the value input in the print statement.
   2. Explain the idea of syntax, and the result of various mistakes.
5. Discuss abstracting in coding. [slide 14]

Homework: Seven things you should know if you're starting out programming

Have students go to the article *Seven things you should know if you're starting out programming* at

http://www.theguardian.com/info/developer-blog/2011/oct/07/programming-developer-journalist (http://www.theguardian.com/info/developer-blog/2011/oct/07/programming-developer-journalist)

Before dividing the students into groups for the next activity, it is important to address the paragraph at the beginning of the reading, which mentions the "coder stereotype" and includes a parenthetical note that the author believes this stereotype to be (largely) accurate. Explain that because of the breadth of application areas of computing in today's world, this stereotype is not an accurate view of the diverse field of computing – there are a wide variety of people in computing and a wide variety of applications of computing education. See videos at http://mcwic.github.io/htmlblocks/computerscientistlibrary.html#top

In pairs or groups of three, assign each group to read and summarize one of the seven programming principles in the article.

1. Logic (not "math")
2. Catch a shooting star (variables)
3. Dictionary (data types)
4. Russian Dolls (things within things, instances)
5. Sausage (processes)
6. The dog, the cat, and the fish (causation, event change)
7. Pizza (abstraction) include why abstraction is "like making pizza," and what other kinds of activities might fall into that category.

Check for Understanding: Each person should write a 140 character tweet on their topic, next class the group should share their findings with the class.

Options for Differentiated Instruction

Oral reading strategies such as "popcorn reading" where students take turns reading a paragraph and then pass the reading off to another student in the class, or other reading strategies such as students reading together quietly in pairs, can be used for longer texts.

Longer readings can be broken up by sections or paragraphs to speed the lesson up or keep students engaged.

Activity 1 could be assigned as homework from the day before as a step into this lesson to allow more time in class for the other readings:

Additional activity: Ask students to read the Preface (pages v-vi) and Chapter 1 introduction and section 1.1 (pages 1-2) of *Python for Everybody* http://www.pythonlearn.com/book.php (http://www.pythonlearn.com/book.php) and answer these questions:

1. Why does the author think that Python is a better teaching language for beginning programmers than Java?
2. Why does the author consider programming to be a creative activity?
3. What things are computers better at doing than people?
4. What things are people better at doing than computers?
5. What are some motivations for writing computer programs? List some responses not included in the selected reading.
6. What do you think of the author's writing style? Name some positive and negative aspects of the presentation in the text.

Evidence of Learning

Formative Assessment

Students will use several different strategies for reading and writing responses based on their readings.

Classroom discussions and student responses (written and oral) will allow instructors to check for understanding.

Summative Assessment

Summative assessment will be included in Part 2 on the use of the PyCharm IDE.

Student handout filled in

Tweet from optional homework assignment

Progress recorded in Runestone for the General Introduction

- The Way of the Program (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/intro-TheWayoftheProgram.html)
- Algorithms (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/Algorithms.html)
- The Python Programming Language (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/ThePythonProgrammingLanguage.html)
- Executing Python in this Book (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/SpecialWaystoExecutePythoninthisBook.html)
- More About Programs (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/MoreAboutPrograms.html)
- A Typical First Program (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/ATypicalFirstProgram.html)

Internal Use Only

Future Work

Possibly create simplified versions of the articles for easier reading if everyone decides it's a good idea (per Meg's comment:

Programming for Good: The Story of Code for India, Lexile is too high 1540L. Needs review for vocabulary & sentence structure

For Seven things you should know if you're starting out programming, Lexile in range 1130L.)

(http://csmatters.org) 2 - 2

0b10 - 0b10

Using Python and PyCharm

Unit 2. Developing Programming Tools

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

PyCharm, an IDE for Python, will be introduced. Keywords, file, and variable naming conventions will be addressed.

Outcomes

- Students will find and launch PyCharm.
- Students will configure PyCharm's appearance.
- Students will create a Python project in PyCharm.
- Students will name and save projects according to the requirements of their instructor.
- Students will create a Python file (.py file).
- Students will add line numbers to the file.
- Students will add comments to the file.
- Students will write, debug, and run a simple Python program.

Overview

1. Getting Started (10 min)
2. Guided Activity (20 min)
    1. Introduction to PyCharm
3. Summative Assessment (20 min)
    1. Programming Exercise
4. Homework Assignment

Learning Objectives

CSP Objectives

- 1.2.1 - Create a computational artifact for creative expression.
    - 1.2.1A - A computational artifact is something created by a human using a computer and can be, but is not limited to, a program, an image, an audio, a video, a presentation, or a Web page file.
    - 1.2.1B - Creating computational artifacts requires understanding of and use of software tools and services.
    - 1.2.1C - Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming integrated development environments (IDEs), spreadsheets, three-dimensional (3-D) printers, or text editors.
- 1.2.5 - Analyze the correctness, usability, functionality, and suitability of computational artifacts.
    - 1.2.5A - The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact.
    - 1.2.5B - A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2D - Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.
    - 5.1.2E - Documentation about program components, such as code segments and procedures, helps in developing and maintaining programs.
    - 5.1.2F - Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments.
    - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
    - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
    - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.

1.2.5 AB

5.1.2 DEFJ

5.4.1 CEF

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases

- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products

Key Concepts

Students will learn what an IDE (Integrated Development Environment) is and why it is good to use one when programming.

Students will be able to find, configure and use the PyCharm IDE to write, save, run, debug and retrieve their Python modules according to the requirements of their instructor.

Essential Questions

- How can computing and the use of computational tools foster creative expression?
- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How do computer programs implement algorithms?
- How do people develop and test computer programs?

Teacher Resources

Student computer usage for this lesson is: **required**

For the Students

- Python 3.4.1:
  - https://www.python.org/downloads/ (https://www.python.org/downloads/)
- Free community edition of PyCharm:
  - http://www.jetbrains.com/pycharm/ (http://www.jetbrains.com/pycharm/)
- Non-Programmer's Tutorial for Python 3/Who Goes There?:
  - http://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3/Who_Goes_There%3F (http://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3/Who_Goes_There%3F)
- An Informal Introduction to Python:
  - https://docs.python.org/3.4/tutorial/introduction.html#using-python-as-a-calculator (https://docs.python.org/3.4/tutorial/introduction.html#using-python-as-a-calculator)

Lesson Plan

Getting Started (10 min)

- **Journal:** What environments have you already used to write Python programs?
- Share answers in class.
- After students have shared their answers, introduce the topic of Integrated Development Environments (IDEs). Explain that an IDE provides a way to manage files, write code, get help with code syntax and usage, and the ability to test, run, and debug programs in an environment they are able to configure to their own liking.
- As defined by Wikipedia: "An **integrated development environment** (**IDE**) or **interactive development environment** is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, automation tools, and a debugger. Most modern IDEs offer intelligent code completion features." (From: http://en.wikipedia.org/wiki/Integrated_development_environment (http://en.wikipedia.org/wiki/Integrated_development_environment))
- Have students go to Python Central at http://www.pythoncentral.io/the-best-python-ides-you-can-use-for-development/ (http://www.pythoncentral.io/the-best-python-ides-you-can-use-for-development/) for a list of the top IDEs for Python. PyCharm is listed as one of the best. Explain to students that they will be learning to use the PyCharm IDE.
- Share tweets from the homework about the 7 aspects of programming quickly standing up in place and sharing.
  1. Catch a shooting star (variables)
  2. Dictionary (data types)
  3. Russian Dolls (things within things, instances)
  4. Sausage (processes)
  5. The dog, the cat, and the fish (causation, event change)
  6. Pizza (abstraction) include why abstraction is "like making pizza," and what other kinds of activities might fall into that category.

If they did not assign the homework, review Seven things you should know if you're starting out programming

Have students go to the article *Seven things you should know if you're starting out programming* at

http://www.theguardian.com/info/developer-blog/2011/oct/07/programming-developer-journalist (http://www.theguardian.com/info/developer-blog/2011/oct/07/programming-developer-journalist)

Activity (20 min)

Introduction to PyCharm

- Find and launch PyCharm. If there is not a shortcut on your desktop, go to Programs >> JetBrains >> JetBrains PyCharm Community Edition 3.4.1.
- The first screen will provide options for Create New Project, Open Directory, Check out from Version Control, Configure, and Docs and How-Tos.
- Select Configure >> Settings >> Appearance. Explore Screens under UI Options. IntelliJ is the default appearance
  - Darcula and Windows are some other options.
  - Students should explore other options available.
  - Students should play with the appearance and select a new one (with your approval) if they have a preference.
- Click the arrow to go back to the Quick Start menu.
- Click Create New Project.
- Give the project a name.
  - Click the ellipses (the … at the end of the location field) to navigate to the location you need to save your work.

Teacher note: This may be a good time to have students configure the default Working Directory as shown in the Python and PyCharm Installation and Configuration Guide file in the lesson resources folder.

- When you click “OK,” your project will be created.
- Click File >> New >> Python File to create your .py file. Name your first .py file **rate_time.**
  - The code for this module will be copied from the ‘Non-Programmer's Tutorial for Python 3/Who Goes There?’ tutorial shortly.
- Click OK to create the file.
  - The user name of the author of the file will appear at the top of the code window with the default comment method.
- Take students through some configuration modifications, such as adding line numbers, showing the console, and changing the default for comment style, all covered in the Installation and Configuration Guide.
  - If you are projecting your screen, you can use Presentation Mode to make viewing easier for students.
- Allow students some time to explore the environment.
- Visit Non-Programmer's Tutorial for Python 3/Who Goes There?
  - http://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3/Who_Goes_There%3F (http://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3/Who_Goes_There%3F)
- Prior to copying the code for rate_time.py and area.py, take students through the sections in the Non-Programmer's Tutorial for Python 3/Who Goes There? on Input and Variables.
- Explain and model variable types and their behaviors, and concatenation.
  - For ease of use, consider using the comma at this time to avoid confusion about concatenation.
- Create, save, run and debug programs rate_time.py and area.py.
- To run a program for the first time, go to **Run** and select the file you want to run.
  - After that, the Run command will appear in the upper right corner of the PyCharm IDE.
- As you add new modules to the project you will need to go to Run and select the new file.
  - After that, the file names will appear in a drop-down menu next to the green Run arrow.
- If it has not already been configured, go to Tools >> Run Python Console to show the console at the bottom of the PyCharm environment. Students can use this to test their code as they have in other environments.
- In the PyCharm console, test code from An Informal Introduction to Python 3.1 Using Python as a Calculator. (https://docs.python.org/3.4/tutorial/introduction.html#using-python-as-a-calculator (https://docs.python.org/3.4/tutorial/introduction.html#using-python-as-a-calculator)).

Summative Assessment (20 min)

Students are to:

- Create a new Python module (.py) called name.py.
- Comment in their name, date, and name of project as directed by their instructor at the top of their code window.
- Assign their full name to an appropriately named variable.
- Assign their age to an appropriately named variable.
- Output their name and age as a sentence, with variables and literal values concatenated correctly.
- Debug any errors and run their program with the correct output and without error.

Sample code (will throw an error)

```
# author = 'iam tester'
 # date July 4, 2024
 # name.py
 name = 'Ima Tester'
 age = 15
 print (name + " is " + age + " years old.")
```

The above code will throw an error, because `age` is an integer and needs to be expressed as a string OR concatenated using a comma `(,)`.

```
name = 'Ima Tester'
age = 15
print (name,"is",age,"years old.")
```

Important notes about naming your files and variables:

- Our Python coding examples will follow the PEP 8 Style Guide. (http://legacy.python.org/dev/peps/pep-0008/ (http://legacy.python.org/dev/peps/pep-0008/))
  - Teachers are not required to use this standard but should establish their classroom conventions for students to follow.
- Modules written in Python, .py file names, should be in all lower case as is done with the sample files in the Python for Informatics textbook. (http://www.py4inf.com/code/ (http://www.py4inf.com/code/))
- Variable names should begin with a lowercase letter with multiple words separated by an underscore.
  - Teachers may choose to use camelCase or another naming system and should be specific and consistent with students regarding these requirements.
- Python file names and variable names must not begin with a number, and cannot be the same as Python keywords:

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

(from http://www.pythonforbeginners.com/basics/keywords-in-python (http://www.pythonforbeginners.com/basics/keywords-in-python))

- For more information concerning variable names and keywords to Python, go to
  - Informatics: Exploring Information, Charles Severance, version 0.0.8 Chapter 2, section 2
  - http://www.pythonlearn.com/book.php (http://www.pythonlearn.com/book.php)

Homework Assignment

1. Use two sources to find definitions of "algorithm", cite the sources
2. Write down the steps involved in a daily task – creating a peanut butter and jelly sandwich; getting to school; brushing teeth; completing homework; etc…)

Options for Differentiated Instruction

Students can work in pairs, side by side, to help each other through each step of the process.

Evidence of Learning

Formative Assessment

Checks for understanding throughout the entire process of learning to use the PyCharm IDE by using active participation in trying each step of the process and having students help their elbow partner when difficulties arise.

Summative Assessment

Assessment tasks:

- Create a new Python program file.
- Add a comment containing their name, date, and name of project as directed by their instructor at the top of their code window.
- Assign their name to an appropriately named variable.
- Assign their age to an appropriately named variable.
- Output their name and age as a sentence with variables and literal values concatenated correctly.
- Debug any errors and run their program with the correct output and without error.

(http://csmatters.org) 2 - 3

0b10 - 0b11

Algorithms: Basics

Unit 2. Developing Programming Tools

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

This is the first day of a two-session lesson sequence with topics covered by mini-lectures, explorations, and practice exercises.

Outcomes

- Students will be able to answer the question, "What is an algorithm?"
- Students will explore algorithms described in English
- Students will understand Magic Square construction and parallel algorithms

Overview

1. Getting Started (5 min)
2. Activities (40 min)
3. Wrap-Up (5 min)

Learning Objectives

CSP Objectives

- 2.2.1 - Develop an abstraction when writing a program or creating other computational artifacts.
  - 2.2.1A - The process of developing an abstraction involves removing detail and generalizing functionality.
  - 2.2.1B - An abstraction extracts common features from specific examples in order to generalize concepts.
  - 2.2.1C - An abstraction generalizes functionality with input parameters that allow software reuse.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
  - 4.1.1H - Different algorithms can be developed to solve the same problem.
- 4.1.2 - Express an algorithm in a language.
  - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
  - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
  - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
  - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
  - 4.1.2I - Clarity and readability are important considerations when expressing an algorithm in a language.

2.2.1 ABC

4.1.1 ABCDEGH

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP7: Look for and make use of structure.

Common Core Math:

- A-SSE.1-2: Interpret the structure of expressions
- A-SSE.3-4: Write expressions in equivalent forms to solve problems

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases

- WHST 12.4 - Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience

NGSS Practices:

- 2. Developing and using models
- 5. Using mathematics and computational thinking
- 6. Constructing explanations (for science) and designing solutions (engineering)

Key Concepts

Students will be expected to learn to provide a definition of "algorithm".

Students will be expected to learn to identify the characteristics of describing algorithms in English, pseudocode, or a programming language.

Essential Questions

- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How are algorithms implemented and executed on computers and computational devices?

Teacher Resources

Student computer usage for this lesson is: **none**

PowerPoint Slides for mini-lectures (AlgorithmsPseudocode1.pptx in the Lesson Resources folder)

Excel sheet summary for creating a 4-by-4 Magic Square (MagicSquare4by4.xls in the Lesson Resources folder)

(optional) 16 papers, each with a number from 1-16 on it for students to use when acting out the magic square.

Lesson Plan

Getting Started (5 min)

- Journal Entry Review

–Small group, then large group, review of two previous homework assignments (students were assigned to use two sources to find definitions of "algorithm" and were assigned to write down the steps involved in a daily task – creating a peanut butter and jelly sandwich; getting to school; brushing teeth; completing homework; etc…) The definitions, sources, and algorithm steps were to have been entered into their journal.

Activities (40 min)

- Guided Activity 1

– Act out selected homework “steps of a daily task” to highlight the potential ambiguity of English instructions.

- Mini-lecture 1

– Review Understanding by Design (UBD)-style slide for “Algorithms and Pseudocode: Need to Understand / Important to Know or Do / Worth Being Familiar With” (slide 2 of AlgorithmsPseudocode1.pptx presentation in Lesson Resources folder).

- Guided Activity 2

– Class creates a 4-by-4 Magic Square, as described in MagicSquare4by4.xls in Lesson Resources folder. Sixteen students should be chosen to represent the sixteen numbers, and physically move into the 16 spaces following the algorithm in this document. (This provides an example of a “simple” algorithm that solves a more complex problem; it also gets the students out of their seats and moving around).

- Mini-lecture 2

–In AlgorithmsPseudocode1.pptx (presentation in Lesson Resources folder), walk through “Main Ideas”; “Representing Algorithms”; and “Sequential Algorithms” slides, including pseudocode circle example.

Wrap Up (5 min)

- Review concept of "algorithm"
- Homework: Assign students to write pseudocode for selected algebraic computations.

Options for Differentiated Instruction

"In your words" pair/share as the concept of what an algorithm is (and what is not an algorithm -- e.g. "sort the numbers") is developed.

Class-wide development of the graphical organizer should be facilitated with scaffolding to support students who are having difficulty with the concept.

Evidence of Learning

Formative Assessment

The following "Checks for Understanding" could be used to guide the students towards the two learning objectives.

Objective: Students will be expected to learn to provide a definition of "algorithm".

1. Example/Non-Example: As the class develops a consensus definition of "algorithm," a list of daily living tasks and technical tasks will be discussed to see if their description can match the definition.
2. Exit Ticket: As students leave the room they will hand the teacher a definition of "algorithm", in their own words, plus an example. The teacher will use these to guide the next day's instruction.

Objective: Students will learn to identify the characteristics of describing algorithms in English, pseudocode, or in a programming language.

1. Graphic Organizer: As the characteristics of English, pseudocode, and a programming language are compared and contrasted, a table will be created on the board that captures the dimensions of what the students discuss. The students will score the result as to how well it helps their understanding of the concept. When it is "very good," we will capture it for their use.

Summative Assessment

Students will research formal definitions of algorithms. These will be entered into their journals and pair-shared with a peer.

Students will write out the sequence of steps in one or more daily living tasks, such as "brushing their teeth" or "building a peanut butter and jelly sandwich." These will be entered into their journals and pair-shared with a peer. Selected solutions will be "acted out" in the classroom.

Internal Use Only

Pending Tasks

Bill will make the revisions suggested by Marie (explaining the algorithm) to the teacher's notes document.

(http://csmatters.org) 2 - 4

0b10 - 0b100

Algorithms: Pseudocode

Unit 2. Developing Programming Tools

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Pre-lesson Preparation

This is the second session on algorithms

Summary

During the second session, the students will use pseudocode to describe an algorithm.

Outcome

- Students will write pseudocode using sequencing, selection, and iteration constructs.
- Students will write pseudocode for algebra / geometry formulas.
- Students will write pseudocode for determining if a year is a leap year.

Overview

1. Getting Started (5 min)
2. Guided Activities (40 min)
    1. PowerPoint
    2. Think-Pair-Share
    3. Solutions check
3. Wrap Up (5 min)

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
    - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.5 - Analyze the correctness, usability, functionality, and suitability of computational artifacts.
    - 1.2.5B - A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact.
    - 1.2.5C - The functionality of a computational artifact may be related to how it is used or perceived.
    - 1.2.5D - The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.
- 2.2.1 - Develop an abstraction when writing a program or creating other computational artifacts.
    - 2.2.1A - The process of developing an abstraction involves removing detail and generalizing functionality.
    - 2.2.1B - An abstraction extracts common features from specific examples in order to generalize concepts.
    - 2.2.1C - An abstraction generalizes functionality with input parameters that allow software reuse.
- 4.1.1 - Develop an algorithm for implementation in a program.
    - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
    - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
    - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
    - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
    - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
    - 4.1.1H - Different algorithms can be developed to solve the same problem.
- 4.1.2 - Express an algorithm in a language.
    - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
    - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
    - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
    - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
    - 4.1.2I - Clarity and readability are important considerations when expressing an algorithm in a language.
- 5.2.1 - Explain how programs implement algorithms.
    - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
    - 5.2.1B - Program instructions are executed sequentially.
    - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
    - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
    - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.

2.2.1 ABC

4.1.1 ABCDEGH

4.1.2 ABCGI

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP7: Look for and make use of structure.

Common Core Math:

- A-SSE.1-2: Interpret the structure of expressions
- A-SSE.3-4: Write expressions in equivalent forms to solve problems
- F-IF.1-3: Understand the concept of a function and use function notation
- F-IF.4-6: Interpret functions that arise in applications in terms of the context

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- WHST 12.4 - Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience

NGSS Practices:

- 2. Developing and using models
- 5. Using mathematics and computational thinking
- 6. Constructing explanations (for science) and designing solutions (engineering)

Key Concepts

Students will write pseudocode using sequencing, selection, and iteration constructs.

Essential Questions

- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How are algorithms implemented and executed on computers and computational devices?
- How do computer programs implement algorithms?

Teacher Resources

Student computer usage for this lesson is: **optional**

Files in the Lesson Resources folder:

AlgorithmsPseudocode2.pptx : PowerPoint Slides for mini-lectures

Student Handout and Key for Matching Pennies Game

Student Handout for Rock Paper Scissors

Psuedocode Summary and Examples of common Algorithms.docx

Lesson Plan

Getting Started (5 min)

- Homework Review from Session 1 (students were assigned to write pseudocode for selected Algebra/Geometry calculations)
- Journal: Describe the algorithm of another student. Is there enough detail to allow somebody to follow the steps?

Guided Activities (40 min)

PowerPoint

Walk through “Selection Statements”; “Iteration / Repetition” slides from the AlgorithmsPseudocode2 file in the Lesson Resources folder.

Things to stress with your students:

1. Whenever you need to store information, it must go into a variable. So think about what variables might be needed when you are creating your algorithm

2. Selection and Iteration statements require conditionals. Identify a conditional as something that returns a True or False answer. If selects the next statement to occur by answering the conditional question as being true or false. I have in the past pointed out the True and Then both start with T so TRUE always does the THEN, wherease Else and False both end with LSE, so when the answer if FALSE, you do the ELSE.

While continues to loop as long as the conditional answer is TRUE. When the conditional answer is false, the algorithm jumps to the statement after the End While.

Guided Activity

During powerpoint, guide students through the Game of Matching Pennies (a student working copy and a solution key is in the Master Teacher Resource folder for this lesson).

Think-Pair-Share

Students work in pairs to create and share their pseudocode. Use the Rock Paper Scissors hand out to have student pairs psuedocode Rock Paper Scissors. If there is time, have groups switch algorithms and critique the algorithm of the other group.

Walk through pseudocode syntax summary handout called **Psuedocode Summary and Examples of common Algorithms.docx** in Lesson Resources folder.

Students work through challenges and check their results against sample solutions.

Wrap Up (5 min)

Review slide: "Why we have leap years."

Homework:

Assign students to create pseudocode for leap years.

Options for Differentiated Instruction

Pairing of students and crossing pairs to form groups of four should be used for the set of exercises that are part of this lesson.

Evidence of Learning

Formative Assessment

Think-Pair-Share

Summative Assessment

Students will write pseudocode for algebra / geometry formulas. These will be entered into their class notes.

Students will write pseudocode for determining if a year is a leap year. This will be entered into their journals.

Internal Use Only

Future Work

It'd be nice to have some examples of appropriate algebra/geometry problems for the beginning of the lesson

(http://csmatters.org) 2 - 5

0b10 - 0b101

Reading Python Code and Debugging

Unit 2. Developing Programming Tools

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary

Students learn simple Python programs and their structure, as well as the basics of debugging.

Outcomes

- Students will identify the parts of a Python program
- Students will describe the history of the computer bug.
- Students will look at debugging techniques and classifying errors.
- Students will compare different kinds of errors

Overview

Session 1:

1. Getting Started (5 min) Discussion: What is a program?
2. Guided Activity (40 min) Exploration
3. Wrap Up (5 min)
4. Homework

Session 2:

1. Getting Started (20 min)
    1. Homework Review [5 min]
    2. Runescape Tutorial [15 min]
2. Activity (20 min)
3. Wrap Up (10 min) Journal
4. Homework

Learning Objectives

CSP Objectives

- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
    - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
    - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
    - 5.1.1D - Additional desired outcomes may be realized independently of the original purpose of the program.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2G - Program development includes identifying programmer and user concerns that affect the solution to problems.
    - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
    - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
    - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.
    - 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
    - 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
    - 5.4.1I - Programmers justify and explain a program's correctness.
    - 5.4.1M - The functionality of a program is often described by how a user interacts with it.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
    - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
    - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.

5.5.1 AD

Math Common Core Practice:

- MP6: Attend to precision.
- MP7: Look for and make use of structure.

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases

NGSS Practices:

- 5. Using mathematics and computational thinking

Key Concepts

Programs can be developed to solve problems (to help people, organizations or society), for creative expression, to satisfy personal curiosity or to create new knowledge.

Essential Questions

- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

For the Students:

- "Is Eliza Human?"
  - Use Python to build a chatbot.
  - printing, user input, variables, if statements, while loops
  - https://groklearning.com/csedweek/ (https://groklearning.com/csedweek/)

CodingBat practice

Useful once functions have been taught

http://codingbat.com/python (http://codingbat.com/python)

Development environments for python

Spyder Python Development Environment:

https://github.com/spyder-ide/spyder (https://github.com/spyder-ide/spyder)

Python IdlePycharm Python Development Environment using Python 3.4.1Runestone: Interactive Python:

http://interactivepython.org/runestone/static/CCPS_Python/index.html (http://interactivepython.org/runestone/static/CCPS_Python/index.html)

Lesson Plan

Session 1

Getting Started (5 min)

Journal Prompt: What is a bug in a computer program?

- See if anybody knows the history, this will be covered in Runestone General Introduction What is Debugging? lesson that follows.

Guided Activity (40 min)

Runestone Tutorial [30 min]

- Jigsaw: Divide students up into 5 groups. Have each group prepare a creative presentation on one of the topics below with visuals or an active component (rap, song, dramatic reading, etc.) to convey the message of that section. Groups have 15 minutes to prepare a 2 minute presentation.

- What is Debugging? (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/WhatisDebugging.html)
- Syntax errors (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/Syntaxerrors.html)
- Runtime Errors (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/RuntimeErrors.html)
- Semantic Errors (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/SemanticErrors.html)
- Experimental Debugging (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/ExperimentalDebugging.html)

- After the presentations all students should
  1. make a tree diagram, name the three types of errors, and give examples of each. Which type of error do you think is the hardest to detect and why?

- **Explore a simple Python program together. [10 minutes]**
  - Teacher demonstrates the results of running eliza.py using Pycharm or Idle
  - Hand out the code for eliza.py on paper to students, demonstrate on the front screen have them mark up the code as indicated on the student handout together as a class discussing the vocabulary below (all of these concepts will be taught in Unit 2, this is just a preview).
    (See the teacher guide for answer key.)
  - Vocabulary: (use the Runestone Glossaries if needed)
    - comment
    - function:
    - parameter
    - input
    - output
    - loop
    - condition
    - variable
  - Have students mark up the code at the bottom of the page using the same key on the student handout

Wrap Up (5 min)

Journal: Use formative assessment Questions 1 - 3:

1. How do you make an output statement in Python?
2. How do you make an input statement?
3. What is a user prompt and what is its purpose?

Homework

Set up your IDE and Python at home (if you can) or use an online Python IDE such as https://repl.it/languages/Python3 (https://repl.it/languages/Python3) or ideone.com . Type up a simple "Hello World" program and get it to run. Bring in evidence that it works or write a few sentences about the issues you are having in trying to install it or write about your experience using an online IDE, or write out the code for Hello World without looking at any notes and report on how easy or hard it was to remember the details. (Be careful not to make any student feel awkward for lack of a home computer they are allowed to install softare on)

Session 2

Getting Started (20 min)

Homework Review [5 min]

Collect handwritten notes or evidence that Python 3 is set up at home or the student is able to use https://repl.it/languages/Python3 (https://repl.it/languages/Python3) or ideone.com. As students work on tutorial, address specific issues with students who had trouble with install.

Runestone Tutorial [15 min]

Complete an introductory tutorial: Runestone Interactive- General Introduction- What is Debugging?: http://interactivepython.org (http://interactivepython.org/) GeneralIntro:

- Formal and Natural Languages (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/FormalandNaturalLanguages.html)
- A Typical First Program (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/ATypicalFirstProgram.html)
- Comments (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/Comments.html)
- Glossary (http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/Glossary.html)

Activity (20 min)

Develop the beginnings of a chat bot where the computer and user introduce themselves to each other. The computer asks a question, the user provides a response and the computer responds back again, including the user input within the response (see https://groklearning.com/csedweek/ (https://groklearning.com/csedweek/) for ideas). Extend: Give your chatbot a personality like a friend, grandfather, therapist, or child.

Wrap Up (10 min)

Journal [10 min]

Formative assessment Questions # 4 - 6:

4) Name several different input devices.

5) How do we comment a Python program? Why do we use comments?

6) In your journal, make a tree diagram, name the three types of errors, and give examples of each. Which type of error do you think is the hardest to detect and why?

Homework

Write code to introduce yourself. Display your name. Greet and ask for three interests. Display the three interests and give a reply like "That's interesting!" Print your code. Next day: Have students introduce one another by “running the code" of a classmate. If you cannot get PyCharm to install, use http://ideone.com/ (http://ideone.com/) or repl.it or create this program by hand on paper.

Options for Differentiated Instruction

Suggested strategies:

- Use red, yellow, green sticky notes at computer for a quick check of understanding.
- Provide guide questions for reading during Runestone interactive tutorial.
- Provide teacher-annotated text excerpts or have students annotate their own excerpts that follow the Runestone interactive.

Evidence of Learning

Formative Assessment

In your journal, make a tree diagram, name the three types of errors, and give examples of each. Which type of error do you think is the hardest to detect and why?

Summative Assessment

Find the errors in an algorithm or in Python statements covered to date

Define what debugging is and give examples of the 3 categories of bugs (syntax, logic(semantic) and runtime)

Describe the differences between programming and debugging.

Internal Use Only

Pending Tasks

Dianne 8/30/14: This lesson needs resources.

Day 1 exploration needs sample programs for a new teacher to demo that are at the right level at this point in the Python learning sequence.

Day 2 activity needs guildelines and a rubric for grading.

Future Work

I think it will be essential to provide some more scaffolding materials for teachers (sample programs and examples of each topic to be taught). The master teachers will probably be OK but our pilot teachers will need more.

Joe's suggestion https://piazza.com/class/hpznrt4f9gd2cj?cid=48 (https://piazza.com/class/hpznrt4f9gd2cj?cid=48)

- Runestone's Quick Introduction to Python (http://interactivepython.org/runestone/static/IntroPythonTurtles/index.html) (chapter 1-5 done in ~40 minutes)
- Runestone's Variables, Expressions and Statements (http://interactivepython.org/runestone/static/CCPS_Python/SimplePythonData/simpledata.html)

(http://csmatters.org) 2 - 6

0b10 - 0b110

Types and Evaluation

Unit 2. Developing Programming Tools

Revision Date: Jul 06, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Pre-lesson Preparation

Create a table of values of all types and print a copy of them on the cardstock for each group. Cut the papers up into individual cards, so each value is on its own card. Place each group's cards in a plastic bag.

Summary

Students are introduced to basic programming vocabulary, including integers, floats, strings, values, and expressions. They will work through a set of guided notes and slides, and, then, be released to explore Python through an independent (or paired) exercise.

Outcomes

- Students will learn that values are fundamental things that the program manipulates.
- Students will program four different types of values: floats, ints, strings, and booleans (briefly).
- Students will understand that expressions are evaluated by Python and result in a value.
- Students will understand that statements are executed by Python and may not result in a value.

Overview

1. Getting Started (5 min)
2. Introduction of Content (10 min)
3. Guided Activity: Type Sort (15 min)
4. Independent Activity (15 min)
5. Wrap Up (5 min)

Learning Objectives

CSP Objectives

- 5.1.3 - Collaborate to develop a program.
- 5.3.1 - Use abstraction to manage complexity in programs.
  - 5.3.1I - Strings and string operations, including concatenation and some form of substring, are common in many programs.
  - 5.3.1J - Integers and floating-point numbers are used in programs without requiring understanding of how they are implemented.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
  - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
  - 5.5.1B - Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.
  - 5.5.1C - Real numbers are approximated by floating-point representations that do not necessarily have infinite precision.
  - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
  - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
  - 5.5.1F - Compound expressions using and, or, and not are part of most programming languages.

5.3.1J

5.5.1A

5.5.1B

5.5.1C

5.5.1D

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP3: Construct viable arguments and critique the reasoning of others.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.
- MP8: Look for and express regularity in repeated reasoning.

Common Core ELA:

- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.8 - Evaluate the hypotheses, data, analysis, and conclusions in a science or technical text
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products

NGSS Practices:

- 2. Developing and using models
- 5. Using mathematics and computational thinking

Key Concepts

- Values are fundamental things that the program manipulates.
- So far, we have seen four different types of values: floats, ints, strings, and booleans (briefly).
- Expressions are **evaluated** by Python and result in a value.
- Statements are **executed** by Python and may not result in a value.

Essential Questions

- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?
- What is the difference between an expression and a statement?
- What are the different data types?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- "Values and Types" slides
- "Exploration Question" document

Team Shake App (https://itunes.apple.com/us/app/team-shake/id390812953?mt=8)

Runestone (http://interactivepython.org/runestone/static/thinkcspy/SimplePythonData/simpledata.html#values-and-data-types): Values and Expressions

Lesson Plan

Getting Started (5 min)

- Write a reflection on the homework from last class to write code to introduce yourself. The program should:
  1. Display your name.
  2. Greet and ask for three interests.
  3. Display the three interests
  4. Give a reply like "That's interesting!"

  Did you run into any errors?

  Do you feel that you can easily write this kind of code?

  Share results with an elbow partner.

Teacher Note*:* Ideally, students are paired with people they don't work with as frequently, in order to promote classroom culture.

Introduction of Content (10 min)

Refer to the PowerPoint called Values and Types in the Lesson Resources folder.

- **Value**:
  - Everything on a computer reduces to numbers
    - Letters are represented by numbers (ASCII codes)
    - Pixel colors are represented using three numbers (red, green, blue)
  - Python tells these numbers apart by the use of **types**
- **Type:**

- A set of **values** and the operations performed on them
  - Examples of operations: +, -, /, *
  - The meaning of these depends on the **type**!

- **Integers** (ints):
  - Values: ... -3, -2, -1,0,1,2,3,4
    - **Integer** literals have no commas (1, 2, 10, 1034453)
  - Operations: +, -, * (multiply), /, ** (to the power of)

- **Floating point** (float):
  - Values: (approximations of) real numbers
    - In Python, a number with a `'.'` is a **float** (ie. 2.0)
    - Without a decimal, a number is an **int** (ie. 2)
  - Operations: +, -, * (multiply), /, ** (to the power of)
  - Exponential notation is useful!

Check for Understanding: Have students write an integer on their paper (check with their elbow partner that it is a number). Ask students to turn that integer into a float with the same numerical value. Talk about going the opposite direction (float to int).

(Example answers: int: 3 float: 3.0 or 3.00 or ...)

- **String** (str):
  - Values: Any sequence of characters within double or single quotes
  - Operations: + (referred to as *catenation* or *concatenation*)
  - Concatenation can only apply to strings
    - `"hello" + "world" = "helloworld"`
    - `"hello" + " world" = "hello world"`
    - `"hello"` + 2 produces an error
- **Boolean** (bool) :
  - Values: True, False
    - Boolean literals are just `True` and `False` (they *must* be capitalized!)
  - Operations: not, and, or
    - not b: True if b is False and False if b is True
    - b and c: True if both b and c are True; False otherwise
    - b or c: True if b is True or c is True; False otherwise
  - **Check for Understanding:** *Hold up two objects for the students to see, then ask the following questions:*
    - I am holding object A **and** object B. True or False?
    - *Drop one of the objects.* I am holding object A **and** object B. True or False?
    - I am holding object A **or** object B. True or False?
  - Often comes from comparing **int** or **float** values
    - Order comparison: `i< j` `i<=j` `i>=j` `i>j`
    - Equality, inequality: `i == j` `i!= j` (`'='` means something else!)

Guided Activity: Type Sort (15 min)

1. Group students (see differentiation for grouping options).
2. Ask teams of students to group the values by types.

Teacher Note: See Differentiation for two variations of this activity.

Individual Activity (15 min)

Studens will complete the Exploration Worksheet which is found in the Lesson Resources folder.

- Version 1:
  - Hand out exploration worksheet and allow students to work individually on their computer in PyCharm IDE or terminal/command.
  - *Make sure students are filling in their expected value before determining the calculated value. They need only fill in the "reason for calculated value" if their two values do not match originally.*
- Version 2:
  - If a student needs a little more practice before moving on, send them to the Runestone (http://interactivepython.org/runestone/static/thinkcspy/SimplePythonData/simpledata.html#values-and-data-types) to complete the Variables and Expressions activity before releasing them to the exploration: http://interactivepython.org/runestone/static/thinkcspy/SimplePythonData/simpledata.html#values-and-data-types

Wrap Up (5 min)

- Exit ticket: Give the value of the Python expression. Also give the name of the operator used.

```
2 + 5
5 * 2
5 ** 2
5 / 2
5 % 2
5 + 2 * 4
5 * 2 - 3
```

- Which of the following Python expressions give an error in Python?

```
"pay attention to details'
' " what's for lunch?", my partner asked'
"What's for lunch?", my partner asked"
```

Options for Differentiated Instruction

Ideas for grouping students:

- Team Shake App (https://itunes.apple.com/us/app/team-shake/id390812953?mt=8) A simple app that allows you to put in your students, decide the number of "teams", and randomly groups students. It will allow you to balance teams based on skill or gender, and will allow you to share teams via facebook or email.
- Require students to go find other people in the room and give them a high-five. They may not high-five the person next to them.
- Group by birthday month (make adjustments to the groups as needed)
- Number off
- Ask students to count up the letters in their first name and determine if they have an odd number of letters or an even number of letters. Find three or four others who also have an even or odd number of letters in their name and form a group.
    - Variation if you have talked about other number systems: Find the number of letters mod the number of groups you need.

Group Activity Variations:

1. Give the students a large piece of colored paper and a glue stick, and allow them to glue the types together for later use in the classroom.
2. Create small, individual sets of cards, and have them glue the cards in to their journals by type.
3. Create multiple sets of values. Hand all groups the first set of values and allow them to organize the values by type. For the remaining sets, have groups compete against one another to see who can sort the values the fastest.
4. Print out Bingo cards of values ahead of time and call for: an integer less than 10 and greater than 5, a String with 3 letters, etc. http://www.teach-nology.com/web_tools/materials/bingo/5/ (http://www.teach-nology.com/web_tools/materials/bingo/5/)

Evidence of Learning

Formative Assessment

Checks for understanding are incorporated throughout the lesson.

Summative Assessment

Exit ticket questions are incorporated into lesson.

(http://csmatters.org) 2 - 7

0b10 - 0b111

Creating and Assigning Variables

Unit 2. Developing Programming Tools

Revision Date: Jun 18, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

Students will learn to manipulate variables and value assignments through an activity in which they must become the variable. By the end of the lesson, they will have identified variables as memory locations. They will also assign, copy, and destroy values in order to perform a swap algorithm and visualize Python's manipulation of variables and values in memory.

Outcomes

- Students will learn about variable manipulation and assignment.
- Students will understand variables representing memory locations.
- Students will be able to use variables in Python.

Overview

1. Getting Started (5 min) - Journal
2. Introduction of Content (25 min) - Visualizing Variables
3. Independent Activity (15 min) - "Swap to the Top"
4. Wrap Up (5 min)

Learning Objectives

CSP Objectives

- 5.1.2 - Develop a correct program to solve problems.
  - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
- 5.1.3 - Collaborate to develop a program.
  - 5.1.3D - Collaboration can make it easier to find and correct errors when developing programs.
  - 5.1.3E - Collaboration facilitates developing program components independently.
- 5.2.1 - Explain how programs implement algorithms.
  - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
  - 5.2.1B - Program instructions are executed sequentially.
  - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.

5.1.2 B

5.1.3 DE

5.2.1 AB

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP3: Construct viable arguments and critique the reasoning of others.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.
- MP8: Look for and express regularity in repeated reasoning.

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- WHST 12.1 - Write arguments on discipline specific content
- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products

NGSS Practices:

- 1. Asking questions (for science) and defining problems (for engineering)
- 5. Using mathematics and computational thinking

Key Concepts

- Variables are memory locations that we have named, so we can put things in those locations.
- Variables contain values, and can only contain one value at a time.
- When assigning one variable from another, a copy is made rather than a transfer.
- If a variable is being assigned after having been initialized to a previous value, that previous value is lost and no longer accessible unless that value has been copied to another variable space.

Essential Questions

- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How are programs developed to help people, organizations or society solve problems?
- How do computer programs implement algorithms?

Teacher Resources

Student computer usage for this lesson is: **none**

For the Students:

- Cups
- Notecards
- String and paper to create large name labels (could also use nametags)
- Worksheet: "Swap to the Top.docx" (in the Lesson Resources folder)
- Homework

Optional:

- Differentiated one-pager (see "Options for Differentiated Instruction" in lesson).

For the Teacher:

- Write the following code on the board (after students have a chance to brainstorm their own solutions):

```
team1 = "Miami Heat"
team2 = "Washingon Wizards"
temp = team1
team1 = team2
team2 = temp
```

Lesson Plan

Getting Started (5 min)

Journal

What is a variable? What are some things in your life that change often?

Introduction of Content (25 min)

Introduction of Variables [5 min]

See slides in Lesson Resources Folder for a guided introduction.

Guided Activity: Visualizing Variables [15 min]

Materials

Disposable cups, index cards, names for variables on strings (to hang around students' necks)

Setup

1. We're about to talk about sports teams and their ranking in terms of variables. When we place teams on a ranking board, Team 1 is always at the top, followed by Team 2, Team 3, etc. Write a ranked set of 3 to 5 (real or invented) teams on the board. As they play one another, this ranking may change and someone new becomes Team 1.
2. Ask for a student volunteer. Give this volunteer a cup: they are now representing a new variable! In order for a computer to know which variable is which, they need names . Variable names can be any sequence (beginning with a letter or underscore, and including no

spaces), but usually we use words that describe what is inside the variable. (For example, if I have a variable for "student's age," I'm not going to name it `tree`, because the word `tree` has no relevance to your age.)

3. Label this variable `team1` because we're talking about sports teams. Put the name around the student's neck to indicate they have become the variable team1. This student has now become `team1`, and should not respond to any other names! *Note:* A variable name will *never* have quotes around it - that would indicate that it is a string (i.e., a type of *value* rather than a variable's *name)*. Variables can be identified (loosely) by words or letters that are not in quotes and are not keywords (for, if, else, etc.).
4. Next, we will *assign* a value to the variable. This means we will store that value in the location `team1`. Because we have never used this variable before, we are *initializing* it - putting something in this box (memory location) for the first time. On the board, write `team1 = "Miami Heat"`
5. **CFU: What is the type of "Miami Heat"?**
   ("Miami Heat" is a string. We know this because it has quotes around it.)
6. The equals sign is what allows us to make this assignment. We are not saying that `team1` is the same as the Washington Wizards; it is only where were are currently keeping that team. Write "Miami Heat" on a notecard and put it into the cup.
7. Next, we need to *initialize* our second variable and *assign* a value to it. Follow the same process as `team1` and "Miami Heat", this time using `team2` and "Washington Wizards".
8. **CFU: What is the difference between a value and a variable?**
   (Values can be stored in variables. The content of a variable can change to different values, but its name will always be the same because the name is just an identifier of a location in memory.)
9. We've now *initialized* our two variables, but as it turns out… last night the two teams played, and the order was upset, putting the Wizards ahead! We have to model a *swap* in our code to reflect this change and fix our scoreboard.
10. **Think-Pair-Share:** A value is only safe (and not lost to the world of cyberspace) if it is in a variable. A variable can only hold one value at a time. How can we swap values between `team1` and `team2`?
    (Allow students 3 minutes to brainstorm.)
11. Call on students and have them direct `team1` and `team2` to swap their values. **Key point:** when we access a variable and put its value in the place of another variable, that value is being copied. By doing so, however, any previous value in the variable is lost**.**
12. Ideally, a student will realize they must create another variable and do so, walking each other through the completion of the swap. If not, you should guide them towards this discovery.
13. **CFU (if necessary): Cold call a student to create a third variable, `temp`.**
    (Make sure to call up a third member of the class to become the variable `temp`.)
14. If students have not at this point finished the swap, continue to walk them through the swap of a value to the `temp` variable, so the variables do not overwrite one another.
15. *Initialize* a variable `temp` (named so because we will not be using it very long). Write `temp = team1` on the board. Notice that in this assignment, it looks like we are setting a variable equal to another variable. Instead, we are setting the variable `temp` equal to the *value* inside the variable `team1`. This means that the value is copied to a second location.
16. Call a new volunteer to the floor. Give them a cup and name them `temp`.
17. Write a second notecard labeled "Miami Heat" and put it in the `temp` cup.
18. Now that the "Miami Heat" value has been saved somewhere, we can swap "Washington Wizards" into `team1`. Write `team1 = team2` on the board.
19. Go to `team2` and copy the text from the notecard onto a new one. Move to `team1` and replace the previous notecard with this new one.
20. A variable can only store one value at the time, which means that we've lost whatever was in `team1`. Rip up the notecard or throw it in the trash.
21. Complete the final assignment of `team2 = temp` by following the same process.

Synopsis

Look at the code written on the board and ask a student to walk us through each step.

CFU: Why did we create the variable `temp`?

(Because variables overwrite the values of one another, and if we were to just set `team2 = team1` we would lose one of the values.)

Students should notice:

- Variables are memory locations that we have named, so we can put things in those locations.
- Variables contain values, and can only contain one value at a time.
- When assigning one variable from another, a copy is made rather than a transfer.
- If a variable is being assigned after having been initialized to a previous value, that previous value is lost and no longer accessible.

Discussion [5 min]

In reality, Python actually has a "shortcut syntax" that allows us to make this swap in one step. It looks like this:

```
a,b = b,a
```

If we wanted to swap the *values* in `team1` with `team2`, we would simply have to write:

```
team1,team2  = team2, team1
```

Here, Python is doing exactly what we were doing. It is internally creating a variable (which has no name but serves the same purpose as our `temp` variable), using it as a place holder, and then completing the swap.

Think-Pair-Share

Why would the makers of Python build in this function? What other uses does it have?

Independent Activity (15 min)

A version of this worksheet can be found in the Lesson Resources folder, titled "Swap to the Top".

Give students a list of games that have been played by the teams on the board, and the resultant new ranking. Have them create a piece of code that will reorganize the teams into the correct ranking. (They can assume that the variables `team1` ... `team13` already exist and have been initialized to the Friday ranking.) Give them the option of doing so through the use of manipulatives, or on their own paper.

Wrap Up (5 min)

See Exit Ticket in Lesson Resources Folder

Journal

What is the difference between a variable in a math class and in a computer science class? What is the difference between a float and an integer? Why would you use one instead of the other?

Homework

Continue working on "Swap to the Top" worksheet.

Options for Differentiated Instruction

Create a 3 column 'one-pager'. In the left column, create a copy of the code from the activity (a swap algorithm) including `team1`, `team2`, and `temp` that performs the complete swap. In the middle column, write steps that occur in the code (Step 1: Initialize variable and assign the value). In the third column, the students should draw a visual of what that looks like in terms of disposable cups and index cards. (They can just write the variable name on the cup, rather than drawing in a person as well.)

Evidence of Learning

Formative Assessment

Checks for Understanding are embedded in the lesson. They are also shared below.

What is the type of "Miami Heat"?

("Miami Heat" is a string. We know this because it has quotes around it .)

What is the difference between a value and a variable?

(Values can be stored in variables. The content of a variable can change to different values, but its name will always be the same because the name is just an identifier of a location in memory.)

Think-Pair-Share: A value is only safe (and not lost to the world of cyberspace) if it is in a variable. A variable can only hold one value at a time. How can we swap values between *team1* and *team2*?

(various answers)

Why did we create the variable `temp` ?

(Because variables overwrite the values of one another, and if we were to just set `team2 = team1`, we would lose one of the values.)

Summative Assessment

Task: Create a piece of code that will rearrange the ranking of the teams in order to reflect the outcome of previous games. (Independent Practice)

Internal Use Only

Pending Tasks

Introduction to Variables powerpoint needs to be created or added to the lesson folder.

Future Work

Complete a better version of "Swap to the Top" worksheet.

Holly suggestion: a more descriptive lesson title would have been immensely helpful during planning time: "How to swap values stored in variables".

(http://csmatters.org) 2 - 8

0b10 - 0b1000

Comparison, Logical Operators, and Conditional Execution

Unit 2. Developing Programming Tools

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

Students will learn how programs can solve problems using the various types of conditional statements in Python programs.

Outcomes

- Students will understand how conditional logic controls program flow.
- Students will be able to solve problems that require conditional logic.

Overview

1. Getting Started (5 min)
2. Introduction of Content (30 min)
    1. Comparison operators [5 min]
    2. Logical operators [5 min]
    3. Conditional statements [20 min]
3. Activity (10 min)
4. Wrap-up (5 min)

Learning Objectives

CSP Objectives

- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
    - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
    - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.
    - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
    - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
    - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.
    - 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
    - 5.4.1I - Programmers justify and explain a program’s correctness.

- 5.4.1L - An explanation of a program helps people understand the functionality and purpose of it.
- 5.4.1M - The functionality of a program is often described by how a user interacts with it.

- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
  - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
  - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
  - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
  - 5.5.1F - Compound expressions using and, or, and not are part of most programming languages.
  - 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.
- MP8: Look for and express regularity in repeated reasoning.

Common Core ELA:

- RST 12.2 - Determine central ideas and conclusions in the text
- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.9 - Synthesize information from a range of sources
- RST 12.10 - Read and comprehend science/technical texts
- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes

Key Concepts

Decisions in programs are made using conditional statements.

Essential Questions

- How are programs developed to help people, organizations or society solve problems?
- How do computer programs implement algorithms?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

How are comparison operators and Boolean expressions used with conditional statements?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- Nested If Statement Setup

Reference Texts:

- RUNESTONE BOOK: How to Think Like a Computer Scientist - http://interactivepython.org/runestone/static/CCPS_Python/Selection/selection.html (http://interactivepython.org/runestone/static/CCPS_Python/Selection/selection.html)
- Python for Informatics: Exploring Informaiton- http://www.pythonlearn.com/book.php (http://www.pythonlearn.com/book.php)

Online interpreters:

- http://www.compileonline.com/execute_python3_online.php (http://www.compileonline.com/execute_python3_online.php)
- http://www.pythontutor.com/visualize.html#mode=edit (http://www.pythontutor.com/visualize.html#mode=edit)
- http://labs.codecademy.com/#:workspace (http://labs.codecademy.com/#:workspace)

Lesson Plan

Getting Started (5 min)

Journal

Think of a decision you make in your daily life and how you make the decision. In your journal, write about your decision and the process you use to decide.

Teaching note: take a few minutes to have students share responses (whole class, elbow partners, small groups).

Introduction of Content (30 min)

Brief discussion of comparison operators [5 min]

==, !=, <, >, >=, <=

Briefly discuss logical operators [5 min]

Includes AND, OR, and NOT

Example: Type various combinations of Boolean values (True / False) with Boolean operators (not, and, or) into the IDE (PyCharm)

Conditional Statements [20 min]

Show physical representation using real-life example

Suggested Activity: What's in the box?

Materials: Two small opaque containers, one small item for each container, two post-it notes.

Set-Up Directions: Set up the activity before the class arrives, following the directions below:

1. Place one item in each container.
2. Using the post-it notes, label one box "green" and the other "black."
3. On the board, write the two colors (green and black) on the board to record the student tally.
4. On the board, write the following pseudocode:

```
if numofGreen > numofBlack:
    open green box
else:
    open black box
```

Activity Directions:

1. Ask: "If you could choose one of these colors, which one would it be? Please only vote once. Raise your hand if you would choose Green." Record the number of students who voted for Green on the board.
2. Ask: "Raise your hand if you would choose Black." Record those votes.
3. Show the students the pseudocode on the board and work through the if/else statement using the data collected from the class.
4. Open the box and display the item that the class voted for.

Discussion of Conditional Execution (if, if/else)

Show the example from How to Think Like a Computer Scientist text (ActiveCode:6 (ch05_4)) or similar example. Briefly demonstrate how to read flowcharts while showing the example from How to Think Like a Computer Scientist.

Check for understanding: Have students answer to the following questions:

1. What will be printed by the following code segment?

```
x=15
if x==25:
  print ('Pizza is yummy')
else:
   print ('My teacher is awesome')
```

2. What will be printed by the following code segment?

```
x=35
y=52
if x!=25 and y==52:
  print ('Pizza is yummy')
else:
 ('My teacher is awesome')
```

Use IDE (PyCharm) to show how to create an if/else statement

Suggested Coding Example:

```
n = input('Please enter your password: ')
if n=='P@s5w0d':
    print ('Welcome, correct user!')
else:
    print ('Incorrect, try again')
```

Activity (10 min)

Journal: Making Predictions [2 min]

Give the students the practice problem from Independent Activity 1 below. Have the students answer the following questions:

1. You are about to create a program using the given criteria. In your opinion, what is the purpose of this program?
2. What do you think the output of this program is when the `food` variable contains the following values?
   pizza
   popcorn

Independent Activity 1 [8 minutes]

If/else practice problem: In the IDE, write a program that will prompt a user to enter a value for a food item. Evaluate the variable `food`. If the value of `food` is equal to "potato salad," display "In Stock". If the value of `food` is not equal to "potato salad," display "Not in Stock".

Test your program with the following values for food:

pizza
popcorn
potato salad

Wrap Up (5 min)

Journal

Thinking about conditional execution, answer the following questions.

1. What difficulties, if any, did you encounter during the development of your program?
2. After running your independent program, were your predictions correct?
3. What conditional execution concepts do you need clarified?

Options for Differentiated Instruction

Alternate Instructional Strategy for Guided Practice :

- For guided practice, have a list of instructions for how to write the example presented.

Alternate Instructional Strategy for Journal: Interactive Journaling

- Comment on the journal entry by asking check-for-understanding questions or clarifying any misconceptions students write about.

Evidence of Learning

Formative Assessment

- A variety of checking for understanding techniques
  - Temperature checks
  - Teacher review student's code
  - Thumbs up/ thumbs down
  - Questioning thoughout the lesson (whole group / small group / individual)
- Quick quizzes
- Interactive journaling

Summative Assessment

- Students create a small program demonstrating conditional execution
- Reflective journal entry

Internal Use Only

Pending Tasks

Need to figure out a consistent and straightforward way to refer to the How to Think... sample code (or replace with an example here, so that teachers don't have to go to the web to get the very simple example!)

Future Work

This lesson would be greatly improved if it contained less contrived examples, which actually connect to something in the real world that the students can understand, and that motivate why you would want to use conditions. The green/black boxes don't seem to mean anything in particular (why would a student vote for one of them? how do the items correspond to the colors?), and the pizza/popcorn example doesn't seem connected to anything at all (why those numbers?)

A teacher could easily have 2 envelopes labelled with the name of 2 sports, or movies, or anything and take a class vote for the favorite of the 2.

(http://csmatters.org) 2 - 9

0b10 - 0b1001

Nested and Chained Conditional Statements

Unit 2. Developing Programming Tools

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Pre-lesson preparation

Students must have an understanding of using conditional statements, and have completed all the assignments from Lesson 3-6 (Conditional Execution (Part 1)).

Summary

Students will learn how programs can solve problems using nested and chained conditional statements in Python programs.

Outcomes

- Students will understand how nested and chained conditional logic controls program flow.
- Students will be able to solve problems that require nested and chained conditional logic.
- Students will work collaboratively to create a small program.
- Students will use their journals to record answers and check for understanding and reflection.

Overview

1. Getting Started (5 min)
2. Introduction of Content (30 min)
    1. Conditional Statements [10 min]
    2. Nested Conditional Statements [15 min]
    3. Chained Conditional Statements [5 min]
3. Activity (10 min)
4. Wrap-up (5 min)

Learning Objectives

CSP Objectives

- 1.2.1 - Create a computational artifact for creative expression.
  - 1.2.1B - Creating computational artifacts requires understanding of and use of software tools and services.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4C - Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation.
  - 1.2.4D - Effective collaboration strategies enhance performance.
- 4.1.1 - Develop an algorithm for implementation in a program.
- 5.1.2 - Develop a correct program to solve problems.
  - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
  - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.
  - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.1.3 - Collaborate to develop a program.
- 5.2.1 - Explain how programs implement algorithms.
  - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
  - 5.2.1B - Program instructions are executed sequentially.
  - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
  - 5.2.1D - An understanding of instruction processing and program execution is useful for programming.
  - 5.2.1E - Program execution automates processes.
- 5.4.1 - Evaluate the correctness of a program.
  - 5.4.1A - Program style can affect the determination of program correctness.
  - 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
  - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
  - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
  - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
  - 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP5: Use appropriate tools strategically.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.

Common Core ELA:

- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.9 - Draw evidence from informational texts to support analysis, reflection, and research

NGSS Practices:

- 1. Asking questions (for science) and defining problems (for engineering)
- 3. Planning and carrying out investigations
- 4. Analyzing and interpreting data
- 5. Using mathematics and computational thinking
- 8. Obtaining, evaluation, and communicating information

Key Concepts

Students must know that conditional statements can be used inside of other condtional statements using nested and chained conditional statements.

Vocabulary:

- nested conditional statements
- chained conditional statements

Essential Questions

- How are programs developed to help people, organizations or society solve problems?
- How do computer programs implement algorithms?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources Folder:

- Nested If Statement Setup

Reference Text:

- How to Think Like a Computer Scientist (http://www.openbookproject.net/thinkcs/python/english2e/) (Ch. Decisions and Selection (http://interactivepython.org/runestone/static/thinkcspy/Selection/selection.html)) - http://interactivepython.org/runestone/static/CCPS_Python/Selection/selection.html (http://interactivepython.org/runestone/static/CCPS_Python/Selection/selection.html)
- Python for Informatics: Exploring Information (http://www.pythonlearn.com/book.php)- http://www.pythonlearn.com/book.php (http://www.pythonlearn.com/book.php)

Project Ideas:

- CS1 Python Programming Projects Archive http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/ (http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/)

Lesson Plan

Getting Started (5 min)

In your journal, write what will be displayed by the following code segment. How do you know your answer is correct?

```
x=2500
if x < 1000:
   print(“small number”)
else:
   print(“big number”)
```

Introduction of Content (30 min)

Brief introduction to conditional statements (10 min)

Suggested Activity: **What’s in the box?** (Part 2)

Show physical representation using real life example by revisiting the box example from previous lesson:

Materials: Four small opaque containers, four small items, two opaque containers big enough to fit two small opaque containers, Six post it notes to label each container with a color

Set Up Directions: Set up the activity before the class arrives following the directions below:

1. Place one small item in each small container
2. With the post-it notes, label the four small containers with different color names (red, orange, yellow, green)
3. Place two labeled containers in each of the larger containers
4. With the remaining post it notes, label one box blue and the other indigo.
5. On the board, write the color names on the board to record the student tally.
6. On the board, write the following pseudocode:

```
if numofBlue > numofIndigo
    if numofRed > NumofOrange
        open red box
    else
        open orange box
else
    if numofYellow > NumofGreen
        open yellow box
    else
        open green box
```

The Activity Directions

1. Ask: “If you could choose one of these colors, which one would it be. Please only vote once. Raise your hand if you would choose Blue. Record the number of students who voted for Blue on the board.
2. Ask: “Raise your hand if you would choose Indigo.” Record the number of students who voted for Indigo on the board.

3. Show the students the pseudocode on the board and work through the if/else statement using the the data collected from the class.
4. Open the the box and display the contents of the box the class voted for.
5. Have students repeat the voting procedure to choose which smaller box to choose and show the object they chose

Brief discussion on nested conditional statements (15 min)

Demonstrate the coding process of the example below.

Guided Activity 1 - Coding Example: Write a program that evaluates a variable "age" and displays a group age category (you are not an adult, you are a senior citizen or you are an adult.)**Suggested Instructional Strategy** - Think Aloud - model your thought process for solving this problem. Include comments in your code.

```
age = 18
if age < 18:
   print(“you are not an adult”)
else:
   if age is > 65:
       print(“you are a senior citizen”)
   else:
       print(“you are an adult”)
```

Brief discussion on chained conditional statements (elif) (5 min)

Guided Activity 2 - Coding Example:

Write a program that prompts the user for their age and displays their age category (you are not an adult, you are a senior citizen or you are an adult.) If they are older than 200 display a message that reads "humanly impossible".

```
age = int(input("How old are you"))
if age < 18:
   print("you are not an adult")
elif age > 200:
   print("humanly impossible")
elif age >= 65:
   print("you are a senior citizen")
else:
   print("you are an adult")
```

Check for understanding

Assign students random numbers from 0 - 250. Have students decide what group they belong to (child, adult, senior citizen, not human) and explain why they made their choice. *** You could have students physically move into the groups, hold up a sign with their answer or answer quietly in their journal.

Activity (10 min)

Collaborative Coding (10 minutes)

1. In your group, develop the pseudocode for the following problem:
   A year is a **leap year** if it is divisible by 4 unless it is a century that is not divisible by 400. Write a function that takes a year as a parameter and returns True if the year is a leap year, False otherwise.
2. In your group, write the program for the problem.

*****Note**

> It is recommended that:
>
> Both partners should be actively involved in the program development. For example, you may choose to employ Pair Programming, in which one partner “drives” (types and uses the mouse) while the other “navigates” (reviews and helps to guide what the driver is doing), with the partners changing roles every 20 minutes. Another method of collaboration is for each partner to develop pieces of the program, combine those pieces, and provide frequent feedback to each other during the development process.

-The College Board Computer Science Principles, Performance Assessment ©2014

Wrap Up (5 min)

Journal

Thinking about Nested and Chained Conditional Statements, answer the following questions:

1. What difficulties, if any did you encounter during development of your program?

2. What difficulties, if any, did you encounter with your partner.
3. If any difficulties did arise, what do you think are some strategies to eliminate the difficulties
4. What concepts in this lesson do you need clarified?

Options for Differentiated Instruction

Extensions

- Have students finish the collaborative coding activity started in class using online collaboration tools such as the ones provided by Google (Google docs, Google Hangout...)
- Have students watch the videos in the How to Think Like a Computer Scientist text book in chapter http://interactivepython.org/runestone/static/CCPS_Python/Selection/selection.html (http://interactivepython.org/runestone/static/CCPS_Python/Selection/selection.html)
- Have students work through the practice problems in the How to Think Like a Computer Scientist http://interactivepython.org/runestone/static/CCPS_Python/Selection/selection.html (http://interactivepython.org/runestone/static/CCPS_Python/Selection/selection.html)
- Assign a practice project from CS1 Python Programming Projects Archive http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/ (http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/)

Homework

- Extension can be assigned as Homework

Alternate Instructional Strategy:

- For guided practice, have a list of list of instructions for how to write the example presented
- Think Aloud - Model your thought process while solving the guided practice problems

Alternate Instructional Strategy: Interactive Journaling

- Comment on the journal entry by asking check for understanding questions or clarifying any misconceptions students write about.

Evidence of Learning

Formative Assessment

- A variety of checking for understanding techniques
  - Temperature checks
  - Teacher review student's code
  - Thumbs up/ thumbs down
  - Questioning throughout the lesson (whole group / small group / individual)
- Quick quizzes
- Peer review
- Interactive journaling

Summative Assessment

Students will create a collaborative program demonstrating concepts introduced in this two part lesson. Students will be assessed using a rubric and will reflect on their learning in their journal

Internal Use Only

Pending Tasks

The pseudocode for leapyears was developed in the algorithms lesson in unit 2. Perhaps we can create a connection, having students use that pseudocode now to turn it into Python code.

(http://csmatters.org) 2 - 10

0b10 - 0b1010

Iteration: For Loops

Unit 2. Developing Programming Tools

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

In this lesson, students will be introduced to the concepts of iteration and `for` loops.

Outcomes

- Students will work through a guided tutorial on `for` loops while being introduced to using turtle graphics in Python.
- Students also have the opportunity to practice writing programs using `for` loops and turtle graphics.
- Students are given the opportunity to use their journal as a reflective tool to make a personal connection between iteration and their personal life.

Overview

1. Getting Started (5 min)
2. Introduction of Content (40 min)
    1. Activity [10 min]
    2. Journal [5 min]
    3. Activity [10 min]
    4. Individual Coding [15 min]
3. Wrap Up (5 min)

Note: Turtle graphic examples in this lesson work with the community version of the PyCharm IDE and Python 3.4.1.

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
    - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
    - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
    - 1.2.3A - Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
    - 1.2.3B - Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision.
- 1.3.1 - Use computing tools and techniques for creative expression.
    - 1.3.1D - Digital effects and animations can be created by using existing software or modified software that includes functionality to implement the effects and animations.
- 4.1.2 - Express an algorithm in a language.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
    - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
    - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.
    - 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
    - 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
    - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
    - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
    - 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP4: Model with mathematics.
- MP5: Use appropriate tools strategically.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.
- MP8: Look for and express regularity in repeated reasoning.

Common Core ELA:

- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.4 - Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience

NGSS Practices:

- 3. Planning and carrying out investigations
- 5. Using mathematics and computational thinking
- 6. Constructing explanations (for science) and designing solutions (engineering)
- 8. Obtaining, evaluation, and communicating information

Key Concepts

Students must understand that programs use the concept of iteration to perform repeated tasks.

Essential Questions

- How can computing and the use of computational tools foster creative expression?
- How can computing extend traditional forms of human expression and experience?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- Turtle Graphics Guided Activity

Required materials:

- 5-10 objects that can be stacked (lego, duplo blocks, plastic cups…) for the first guided activity

Useful additional resources:

- How to Think Like a Computer Scientist (Ch. Python Turtle Graphics (http://interactivepython.org/runestone/static/thinkcspy/PythonTurtle/intro-HelloLittleTurtles.html)) - http://interactivepython.org/runestone/static/thinkcspy/PythonTurtle/intro-HelloLittleTurtles.html (http://interactivepython.org/runestone/static/thinkcspy/PythonTurtle/helloturtle.html)
- How to Think Like a Computer Scientist (Ch. Iteration Revisited (http://interactivepython.org/runestone/static/thinkcspy/MoreAboutIteration/intro-IterationRevisited.html))- http://interactivepython.org/runestone/static/thinkcspy/MoreAboutIteration/intro-IterationRevisited.html (http://interactivepython.org/runestone/static/thinkcspy/MoreAboutIteration/intro-IterationRevisited.html)
- Python for Everybody: Exploring Data in Python 3 (http://www.pythonlearn.com/book.php)
- CS1 Python Programming Projects Archive http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/ (http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/)

Lesson Plan

Getting Started (5 min)

Journal: Think about events in your life that require a repeated action. They could be something simple such as eating a bowl of cereal. List two events in your life that require an action to be repeated. What is the action? What prompts the need for the action to happen? How often does the action happen?

Note: Students will extend their reflections later in the lesson.

Introduction of Content (40 min)

Guided Activity: Physical representation of iteration [10 min]

Materials: 5-10 objects that can be stacked (lego, duplo blocks, plastic cups…)

The activity: Place the objects on a table.

1.Say: “At the conclusion of this activity, all of the objects will be stacked.”

Chose one object to begin with.

2. Say: “I will start with this object and I will continue to stack until there are no single objects left on the table. How many times do you think I will stack an object? Why?”

3. Ask: "Are there any single objects on the table?”

Students should answer yes. Stack one object on your beginning object

4. Ask “ Are there any single objects on the table?”

Students should answer yes. Stack one more object on your started stack. Continue to ask if there are any single objects on the table until the stack is completed and there are no more single objects on the table. Keep a tally of how many times you repeated the process.

5. Ask: “How many times did we repeat the process? Did your prediction match the result?"

6. Explain to the students how the activity represents the concept of iteration and continue into the discussion of iteration.

Journal [5 min]

Choose one of the events you wrote about in your previous journal entry. Take a moment to write the pseudocode for the repetitive action associated with that event.

Note: Check for understanding while students are working.

Example: Eating a slice of pizza:

```
While pizza on plate
    pick up from plate
    take a bite
    place on plate
    bite is consumed
    loop
```

Guided Activity (for loops, how for loops use lists, range, turtle graphic) [10 min]

This guided activity introduces students to `for` loops using turtle graphics.

(See handout in Lesson Resources folder: Turtle Graphics Guided Activity: The for loop)

Individual Coding Activity [15 min]

Give students the following code stem. Have the students alter the code to perform the listed tasks.

Code Stem:

```
import turtle    # Allows us to use the turtles library
window = turtle.Screen()  # Creates a window to display graphics
bob = turtle.Turtle()   # creates a turtle named bob
#Write your code here
window.exitonclick()      # Exits the window when clicked
```

1. Have the turtle draw a triangle using a turtle

2. Now that you know how to add turtles and program them to draw lines repeatedly, use your imagination and creative ability to create your own picture using multiple `for` loops and turtles.

Wrap Up (5 min)

Journal: In your journal summarize the process you used to create your picture. What problems did you encounter? What concepts do you need clarified?

Options for Differentiated Instruction

Students can be given a copy of the guided activity handout to follow along.

Evidence of Learning

Formative Assessment

a variety of checking for understanding techniques

- temperature checks
- teacher review student's code
- thumbs up/ thumbs down
- questioning throughout the lesson (whole group / small group / individual)

quick quizzes

peer review

interactive journaling

Summative Assessment

Students will use for loops and turtle graphics to create graphic representations of iteration. They modify a code stem using turtle graphics to:

1. draw a triangle
2. draw a picture using multiple for loops and turtles.

(http://csmatters.org) 2 - 11

0b10 - 0b1011

Iteration: While Loops

Unit 2. Developing Programming Tools

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

Students continue learning about iteration by using `while` loops and nested iteration. Students work through a guided tutorial on `while` loops and learn more turtle graphics features. They also have the opportunity to practice collaboratively writing programs using `for` loops, `while` loops, and turtle graphics. Throughout the lesson, students are given the opportunity to use their journal as a reflective tool.

Outcomes

- Students will know that iteration can be performed in Python using `while` loops.
- Students will understand that `for` and `while` loops can be written (nested) inside of other `for` and `while` loops.

Outline

1. Getting Started (5 min)
2. Guided Activity (15 min) - `for` and `while` loops with turtle graphics
3. Collaborative Activity (25 min) - Shared coding of an iterative program
4. Wrap Up (5 min) - Journal Activity

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.1 - Create a computational artifact for creative expression.
  - 1.2.1E - Creative expressions in a computational artifact can reflect personal expressions of ideas or interests.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
  - 1.2.3A - Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4D - Effective collaboration strategies enhance performance.
  - 1.2.4E - Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.
  - 1.2.4F - A collaboratively created computational artifact can reflect personal expressions of ideas.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
  - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
  - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
  - 4.1.1F - Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.
  - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
  - 4.1.1H - Different algorithms can be developed to solve the same problem.
  - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
  - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
  - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
  - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
  - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
  - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
  - 5.1.1C - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution.
  - 5.1.1D - Additional desired outcomes may be realized independently of the original purpose of the program.
- 5.1.2 - Develop a correct program to solve problems.
- 5.1.3 - Collaborate to develop a program.
- 5.2.1 - Explain how programs implement algorithms.
  - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
  - 5.2.1B - Program instructions are executed sequentially.
  - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
  - 5.2.1D - An understanding of instruction processing and program execution is useful for programming.
  - 5.2.1E - Program execution automates processes.
- 5.4.1 - Evaluate the correctness of a program.
  - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
  - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.
  - 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
  - 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
  - 5.4.1I - Programmers justify and explain a program's correctness.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
  - 5.5.1A - Numbers and numerical concepts are fundamental to programming.

- 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
- 5.5.1F - Compound expressions using and, or, and not are part of most programming languages.
- 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP5: Use appropriate tools strategically.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.
- MP8: Look for and express regularity in repeated reasoning.

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.9 - Synthesize information from a range of sources
- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes

NGSS Practices:

- 3. Planning and carrying out investigations
- 5. Using mathematics and computational thinking
- 6. Constructing explanations (for science) and designing solutions (engineering)
- 8. Obtaining, evaluation, and communicating information

Key Concepts

- Students will know that iteration can be performed in Python using `while` loops.
- Students will understand that `for` and `while` loops can be written (nested) inside of other `for` and `while` loops.

Essential Questions

- How can computing and the use of computational tools foster creative expression?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- Guided activity: "Turtle Graphics Guided Activity: The while looop" in the Lesson Resources folder

Other:

- How to Think Like a Computer Scientist (http://www.openbookproject.net/thinkcs/python/english2e/) (Ch. Decisions and Selection (http://interactivepython.org/runestone/static/thinkcspy/Selection/selection.html)) - http://interactivepython.org/runestone/static/CCPS_Python/Selection/selection.html (http://interactivepython.org/runestone/static/CCPS_Python/Selection/selection.html)
- CS1 Python Programming Projects Archive http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/ (http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/)

Lesson Plan

Warm Up (5 min)

What will be displayed at the end of this program?

```
y=0
for x in range (0, 8):
    y += 1
print (y)
```

Introduction of Content (15 min)

- Use the guided activity ("Turtle Graphics Guided Activity: The while looop" in the lesson folder) to discuss while loops and nested iteration.
- To save time, have these programs open in your IDE before class starts.

Collaborative Coding Activity (25 min)

Have students work together to complete a coding activity that uses `while` loops, `for` loops, and/or nested iteration. Ideally, the students should design this program from scratch. They should think about a problem to be solved and select the appropriate iteration structure.

Wrap Up (5 min)

Journal: In your journal, summarize the process that you used with your classmates to create the collaborative project. What problems did you encounter? What concepts do you need clarified?

Options for Differentiated Instruction

- Give students a copy of the guided activity instructions so they can follow along.
- Possibly give students a code stem to work from for the collaborative project (while ensuring that they have the opportunity to problem-solve by thinking about and implementing an appropriate loop structure).

Evidence of Learning

Formative Assessment

Various checking-for-understanding techniques:

- Temperature checks
- Teacher reviews of student's code
- Thumbs up/ thumbs down
- Questioning throughout the lesson (whole group / small group / individual)

Quick quizzes

Peer review

Interactive journaling

Summative Assessment

Students will work collaboratively to develop a program that uses nested iteration and turtle graphics.

Internal Use Only

Pending Tasks

Provide the code for the python programs to be demonstrated in the guided activity.

Write a specific Python project for the collaborative coding activity that uses while loops and nested iteration. Need rubric and guidelines for the program to be created.

(http://csmatters.org) 2 - 12

0b10 - 0b1100

Functions: Parameters and Return Values

Unit 2. Developing Programming Tools

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

In Python, a function is a named sequence of statements that belong together. In this lesson, students learn why functions are used, how they are used, and how they are defined.

Outcomes

- Students learn how functions affect program flow, how functions use local variables, and how they use global variables.
- Students learn preferred means of communication with functions and how to incorporate them into their programs. Attention is also given to debugging programs that contain functions.

Overview

1. Getting Started (5 min)
2. Guided Activities (35 min)
    1. Arguments and return values [5 min]
    2. Compound Functions [5 min]
    3. Function Definitions [5 min]
    4. Return Values [10 min]
    5. Line Numbers [10 min]
3. Wrap Up (10 min)

Source

Students will work extensiviely with the online version of How to Think Like a Computer Scientist (HTLACS) hosted by Runestone Interactive.

Learning Objectives

CSP Objectives

- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
    - 1.2.3A - Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
    - 1.2.3B - Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision.
- 2.2.1 - Develop an abstraction when writing a program or creating other computational artifacts.
    - 2.2.1A - The process of developing an abstraction involves removing detail and generalizing functionality.
    - 2.2.1C - An abstraction generalizes functionality with input parameters that allow software reuse.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
    - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
    - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.3.1 - Use abstraction to manage complexity in programs.
    - 5.3.1A - Procedures are reusable programming abstractions.
    - 5.3.1B - A procedure is a named grouping of programming instructions.
    - 5.3.1C - Procedures reduce the complexity of writing and maintaining programs.
    - 5.3.1D - Procedures have names and may have parameters and return values.
    - 5.3.1E - Parameterization can generalize a specific solution.
    - 5.3.1F - Parameters generalize a solution by allowing a procedure to be used instead of duplicated code.
    - 5.3.1G - Parameters provide different values as input to procedures when they are called in a program.
    - 5.3.1M - Application program interfaces (APIs) and libraries simplify complex programming tasks.
    - 5.3.1N - Documentation for an API/library is an important aspect of programming.
    - 5.3.1O - APIs connect software components, allowing them to communicate.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1A - Program style can affect the determination of program correctness.

- 5.4.1B - Duplicated code can make it harder to reason about a program.
- 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
- 5.4.1D - Longer code segments are harder to reason about than shorter code segments in a program.
- 5.4.1E - Locating and correcting errors in a program is called debugging the program.
- 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.

- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.

1.2.3 AB

2.2.1 AC

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP5: Use appropriate tools strategically.

Common Core Math:

- F-IF.1-3: Understand the concept of a function and use function notation
- F-IF.4-6: Interpret functions that arise in applications in terms of the context
- F-BF.3-5: Build new functions from existing functions

Common Core ELA:

- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.10 - Read and comprehend science/technical texts

NGSS Practices:

- 5. Using mathematics and computational thinking

Key Concepts

Students will understand the purpose of functions and how they allow a program to be built and maintained in a modular way.

Predictable Misunderstandings:

- students often think that functions (or any code) written will run. They don't connect that it must be called in order to be run.

- students often think that functions have access to variables that they don't have access.

Essential Questions

- How can computing and the use of computational tools foster creative expression?
- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How are algorithms implemented and executed on computers and computational devices?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?
- How does computing enable innovation?

Teacher Resources

Student computer usage for this lesson is: **required**

Other:

- Function Video
  - http://interactivepython.org/runestone/static/CCPS_Python/Functions/functions.html (http://interactivepython.org/runestone/static/CCPS_Python/Functions/functions.html)
- Python for Informatics: Exploring Information
  - http://www.pythonlearn.com/book.php (http://www.pythonlearn.com/book.php)
- Runestone Interactive
  - http://interactivepython.org/runestone/static/CCPS_Python/index.html (http://interactivepython.org/runestone/static/CCPS_Python/index.html)

Lesson Plan

Getting Started (5 min)

1. Show first 3:30 of Function Video: http://interactivepython.org/runestone/static/CCPS_Python/Functions/functions.html (http://interactivepython.org/runestone/static/CCPS_Python/Functions/functions.html)
2. Students respond in their journals to the following prompts.

- What is a function?
- What is a parameter?
- What are two types of parameters?

Guided Activities (35 min)

Arguments and return values [5 min]

- Students read Python for Informatics: Exploring Informationd Chapter 4 Functions sections 4.1 to 4.3.
- Students copy the sentences below. Replace the words *argument* and *result* with synonyms.
  - “It is common to say that a function "takes" an argument and "returns" a result. The result is called the **return value**.”
- Visit this web page (https://docs.python.org/2/library/functions.html (https://docs.python.org/2/library/functions.html)) that lists the built in functions in Python.
- Verify that the Python Version is 2.7.
- Ask, "How many built in functions are in Python 2.7?"
- Change the Python version to 3.4.1. How many built-in functions are in Python 3.4.1?
- Why do you think there are a different number of functions in Python 3.4.1 than in 2.7?

Compound Functions [5 min]

- Open Runestone Interactive Functions page (http://interactivepython.org/runestone/static/CCPS_Python/Functions/functions.html (http://interactivepython.org/runestone/static/CCPS_Python/Functions/functions.html)) and read about user defined functions in Python.
- Read through lines 1 and 2 that define the header and body of compound statements. Note the format of the compound statement used to define a function.
- Ask:
  - What color is used to highlight the keyword def?
  - What punctuation mark ends the first(header) line?
  - How far is the body indented?
  - What color is the name of the function?
  - What word is used for the value in parenthesis?

Function Definitions [5 min]

- Run ActiveCode 1.
- Change the number in line 15 and run the code to make a smaller square.
- Change the number in line 15 and run the code to make a larger square.
- Run ActiveCode 2.
- Ask: What are the two line numbers used to call or invoke the function drawSquare?
- Run ActiveCode 3.
- Change the definition of drawMulticolorSquare to different colors.
- Ask: What line would you change to draw smaller multi-color squares?

Return Values [10 min]

- Some functions find and return values. Complete ActiveCode 5, 6 and 7.
- Change ActiveCode 7 so it prints the minimum values.

Line Numbers [10 min]

- Run CodeLens 1.
- Ask:
  - What line is executed next after line 6?
  - What line is executed next after line 3?
  - How do you think Python knew what line to go to after line 3?
  - What keyword is used to tell Python to return a value from the function named square?
  - What do you think would happen if you run the program after removing the return statement from line 3?

Wrap Up (10 min)

- Return to Python for Informatics: Exploring Information Chapter 4 Functions. Copy and paste the code from section 4.4 to Runestone ActiveCode 4.
- Run the code multiple times.
- Ask:
  - What do you note about the results?
  - What is the name of the function used?
  - Is the function on of the build in Python functions or is it defined in the program?
  - Where do you think the function is defined?
- Add the number 1 as a parameter in the function call and run the program.
- Explain the error message.
- Write one thing to remember about functions on an exit note.

Homework: Students should read Python for Informatics Chapter 4: Sections 4.7, 4.8 and 4.9. Complete Exercise 2 and 3.

Answer the following questions:

- What error message occured in Exercise 3?
- Why did the error occur?
- What happened when you change the order of print_lyrics and repeat_lyrics function definitions in Exercise 4?
- How is a function call like a road detour?
- The parameter used to define the function print_twice is named bruce. How would the function behavior change if the name was changed to wayne all three times it appears in the function definition?

Options for Differentiated Instruction

Students should be paired through this exercise since paired discussion is used for formative assessment. Question can be provided to students through a variety of formats including production of a Google form or using student response systems.

Three sugested strategies are:

- provide guide questions for reading
- provide teacher provided annotated text excerpts
- Students annotate text while reading

Evidence of Learning

Formative Assessment

After Activity D, Have students compare results with their elbow partners. Discuss any unresolved issues. This strategy can be used after any HTLACS check for understanding.

After Activity F, ask students to suggest a rule for creating functions that would help avoid this error.

Students work alternately between the web site, partners, and the whole group. Teachers are to monitor student responses to the questions following each activity to be sure that students are addressing the key content within each activity.

Summative Assessment

- Explain the advantages of functions/procedures in programming.
- Explain the purpose of parameters with the use of functions.
- Define a Python function with and without parameters.
- Call a Python function with and without parameters.
- Trace a function call from the main program to and from a user defined function.
- Modify a Python program to use a function.
- Create and use functions that return return a string or a number.

(http://csmatters.org) 2 - 13

0b10 - 0b1101

Algorithms: Layers of Abstraction

Unit 2. Developing Programming Tools

Revision Date: Aug 23, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

This is the third session of a three-session lesson sequence with four topics covered by mini-lectures, explorations, and practice exercises.

Outcomes

- Students will translate sample pseudocode into a Python function.
- Students will recognize layers of abstraction for solving a Rubik's cube.
- Students will identify sequencing, selection, and iteration elements in a problem solution.

Overview

1. Getting Started (5 min)
2. Activities (40 min)
    1. Guided Activity 1 [20 min]
    2. [Optional] Mini-Lecture 1 [20 min]
    3. [Optional] Guided Activity 2 [20 min]
    4. Mini-Lecture 2 [20 min]
3. Wrap-up (5 min)

Learning Objectives

CSP Objectives

- 2.2.1 - Develop an abstraction when writing a program or creating other computational artifacts.
    - 2.2.1A - The process of developing an abstraction involves removing detail and generalizing functionality.
    - 2.2.1B - An abstraction extracts common features from specific examples in order to generalize concepts.
    - 2.2.1C - An abstraction generalizes functionality with input parameters that allow software reuse.
- 2.2.2 - Use multiple levels of abstraction to write programs.
    - 2.2.2A - Software is developed using multiple levels of abstractions, such as constants, expressions, statements, procedures, and libraries.
- 2.2.3 - Identify multiple levels of abstractions used when writing programs.
    - 2.2.3B - High-level programming languages provide more abstractions for the programmer and make it easier for people to read and write a program.
- 4.1.1 - Develop an algorithm for implementation in a program.
    - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
    - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
    - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
    - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
    - 4.1.1E - Algorithms can be combined to make new algorithms.
    - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
    - 4.1.1H - Different algorithms can be developed to solve the same problem.
    - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
    - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
    - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
    - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
    - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
    - 4.1.2I - Clarity and readability are important considerations when expressing an algorithm in a language.
- 5.2.1 - Explain how programs implement algorithms.
    - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
    - 5.2.1B - Program instructions are executed sequentially.

- 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
- 5.2.1D - An understanding of instruction processing and program execution is useful for programming.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP7: Look for and make use of structure.

Common Core Math:

- G-CO.6-8: Understand congruence in terms of rigid motions
- G-GMD.4: Visualize relationships between two-dimensional and three-dimensional objects

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.8 - Evaluate the hypotheses, data, analysis, and conclusions in a science or technical text
- WHST 12.4 - Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience

NGSS Practices:

- 2. Developing and using models
- 5. Using mathematics and computational thinking
- 6. Constructing explanations (for science) and designing solutions (engineering)

NGSS Content:

- HS-ETS1-2. Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.

Key Concepts

An algorithm is more than just a sequence of steps: levels of abstraction are crucial to the working of algorithms, and sequencing, iteration, and other control structures are ubiquitous.

Essential Questions

- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How are algorithms implemented and executed on computers and computational devices?
- How do computer programs implement algorithms?

Teacher Resources

Student computer usage for this lesson is: **optional**

In Lesson Resources folder:

- AlgorithmsPseudocode3.pptx : PowerPoint Slides for mini-lectures
- ConwayDoomsdayAlgorithm : file for "Day of the Week" algorithm, Conway's "Doomsday"

[Optional topic] Wikipedia article on Conway date algorithm.

Access to Youtube videos of people and Lego Robots solving Rubik's cubes.

Videos:

- CUBESTORMER 3 Smashes Rubik's Cube Speed Record by ARMflix

  https (https://www.youtube.com/watch?v=X0pFZG7j5cE):// (https://www.youtube.com/watch?v=X0pFZG7j5cE)www.youtube.com/watch?v=X0pFZG7j5cE (https://www.youtube.com/watch?v=X0pFZG7j5cE) and /or

- CUBESTORMER 2 by ARMflix

  https (https://www.youtube.com/watch?v=_d0LfkIut2M)://www.youtube.com/watch?v=_ (https://www.youtube.com/watch?v=_d0LfkIut2M)d0LfkIut2M (https://www.youtube.com/watch?v=_d0LfkIut2M)

- Human World Record: Kopie Van Mats Valk Official Rubik's Cube Single 5:55 by Mats Valk

  https://www.youtube.com/watch?v=UXrxRqwAUkA (https://www.youtube.com/watch?v=UXrxRqwAUkA)

One or more Rubik's cubes.

Links to PDF copies of youcandothecube's solution to Rubik's cube (also copied into the Lesson Resources folder):

You Can Do the Rubik's Cube

- www.youcandothecube.com (http://www.youcandothecube.com/)
- http://www.youcandothecube.com/secret-unlocked/solution-stage-one.aspx (http://www.youcandothecube.com/secret-unlocked/solution-stage-one.aspx)

Lesson Plan

Getting Started (5 min)

Daily Homework Review

- Review of pseudocode for determining if a year is a Leap Year

Guided Activities (40 min)

- Guided Activity 1
  - Demonstrate Python function implementation for the "Is it a Leap Year" pseudocode.
- [Optional] Mini-lecture 1
  - Teach Conway "Day of the Week" algorithm. Use the file called ConwayDoosmdayAlgorithm in the Lesson Resources folder.
- [Optional] Guided Activity 2
  - Have students practice the Conway algorithm.
- Mini-lecture 2
  - Walk through “Extended Example (Rubik's Cube) slides with associated videos; web links; and "props” (actual cubes or website:
    - Coolmath-Games.com Rubik's Cube Game http://www.coolmath-games.com/0-rubikscube/ (http://www.coolmath-games.com/0-rubikscube/) or
    - Google's Rubik's Cube Game https://www.google.com/logos/2014/rubiks/rubiks.html (https://www.google.com/logos/2014/rubiks/rubiks.html)

Wrap Up (5 min)

- Review Table of Contents of selected Algorithms book.
- Review summary slide.
- **Homework:** Take one of the youcandothecube.com solution stages. Identify the sequencing, selection, and iteration elements. Sketch a flowchart of that stage of the solution.

Options for Differentiated Instruction

For the optional activity (Conway Algorithm), some students may have difficulty adding and subtracting dates to translate from a known day-of-the-week to another day in the same month. A chart on the wall, or a current calendar, could be a help.

Some students who are strong in other areas will have difficulty with the spatial aspects of manipulating a cube while retaining an orientation that will let them complete the steps of one of the sub-algorithms without errors. They may need to be paired with another student or the instructor until they master the technique of holding the cube fixed while rotating a face.

The notation of face turning (e.g. R versus R' or L versus L') can be confusing. Having the students practice with an empty jar with a lid can help. Orient the lid up (U), down (D), left (L), right (R), front (F), or back (B). The hand movement to screw the lid on is the same hand movement needed to perform the non-accented face turn. The hand movement needed to screw the lid off is the same as the accented turn (U', D', L', R', F', B').

Variation for class that does not have Rubik's cubes: Use the images from the Solution Guide: www.youcandothecube.com (http://www.youcandothecube.com)

Online rubik's cube solver:

- Coolmath-Games.com Rubik's Cube Game http://www.coolmath-games.com/0-rubikscube/ (http://www.coolmath-games.com/0-rubikscube/) or
- Google's Rubik's Cube Game https://www.google.com/logos/2014/rubiks/rubiks.html (https://www.google.com/logos/2014/rubiks/rubiks.html)

Evidence of Learning

Formative Assessment

The following "Checks for Understanding" could be used to guide the students towards the three learning objectives.

Objective: SWBAT translate sample pseudocode into a Python function.

1. Word Sort. Students will evaluate isolated Python constructs as matching sequencing, selection, or iteration elements.
2. Timed Pair Share. Pairs of students will translate the challange examples from pseudocode into Python.
3. Project Study Group. Looking at pseudocode and non-working Python code: What is wrong? What is missing? How would you change it?

Objective: SWBAT recognize layers of abstraction for solving a Rubik's cube.

1. Describe it. Students will first study and then will identify characteristics of Rubik's cubes (number of sides; possible movements; number and types of pieces -- center, edge, corner; ...)
2. Predict and share -- students will predict strategies for solving the cube.
3. Put in order -- groups of students will take a set of partially solved cubes and place them in "solution order". They will justify their choice. [Note: images from www.youcandothecube.com (http://www.youcandothecube.com) can be used if real cubes are not available.]
4. Learn one layer. Every student will learn how to execute one full step of the solution sequence.

SWBAT to identify sequencing, selection, and iteration elements in a problem solution.

1. Students will identify the sequencing, selection, and iteration steps of: their homework algorithm; two of the sample problems from the challenge set; and one of the stages of the Rubric cube solution. These will be done in pairs and small groups.

Summative Assessment

Students will translate prior pseudocode into Python routines. They will recognize if their programs work correctly.

Students will deconstruct one of the Rubik's cube solution stages. These analysis results will be shared and critiqued.

Internal Use Only

Future Work

I'm not sure that this lesson really explains the abtraction in the algorithms as stated in the objectives. It does continue to develop the complexity of algorithms from the first 2 parts of the algorithm lessons.

(http://csmatters.org) 2 - 14

0b10 - 0b1110

Functions: Scope and Abstraction

Unit 2. Developing Programming Tools

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary

In the first portion of this lesson, students continue their inquiry into the properties of functions, with a focus on communication to and from other functions. In the lab portion of the lesson, students develop three Python modules using both Runestone Interactive and their Python IDE. Students use their own functions to perform calculations and draw a variety of polygons and a circle using turtle graphics.

Outcomes

- Students will understand the process of communication with functions.
- Students will be able to create and call functions.

- Students will be able to use both "fruitful functions" (sometimes just called *functions*) and "unfruitful functions" (which do not return a value - these are usually called *procedures*).
- Students will understand how functions simplify a project's creation.

Overview

Session 1

1. Warmup (10 min)
2. Exploring Functions (40 min)

Session 2

1. Function Labs (45 min)
2. Wrap-up (5 min)

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
  - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
  - 1.2.3A - Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
  - 1.2.3B - Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4C - Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation.
  - 1.2.4D - Effective collaboration strategies enhance performance.
- 5.1.2 - Develop a correct program to solve problems.
  - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
  - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
  - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
  - 5.1.2H - Consultation and communication with program users is an important aspect of program development to solve problems.
  - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.1.3 - Collaborate to develop a program.
  - 5.1.3A - Collaboration can decrease the size and complexity of tasks required of individual programmers.
  - 5.1.3E - Collaboration facilitates developing program components independently.
  - 5.1.3F - Effective communication between participants is required for successful collaboration when developing programs.
- 5.3.1 - Use abstraction to manage complexity in programs.
  - 5.3.1A - Procedures are reusable programming abstractions.
  - 5.3.1B - A procedure is a named grouping of programming instructions.
  - 5.3.1C - Procedures reduce the complexity of writing and maintaining programs.
  - 5.3.1D - Procedures have names and may have parameters and return values.
  - 5.3.1E - Parameterization can generalize a specific solution.
  - 5.3.1F - Parameters generalize a solution by allowing a procedure to be used instead of duplicated code.
  - 5.3.1G - Parameters provide different values as input to procedures when they are called in a program.

Math Common Core Practice:

- MP4: Model with mathematics.
- MP5: Use appropriate tools strategically.

Common Core ELA:

- RST 12.2 - Determine central ideas and conclusions in the text
- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.10 - Read and comprehend science/technical texts

NGSS Practices:

- 1. Asking questions (for science) and defining problems (for engineering)
- 5. Using mathematics and computational thinking

Key Concepts

- Students will understand the process of communication with functions.
- Students will be able to create and call functions.
- Students will be able to use both fruitful and unfruitful functions.
- Students will understand how functions simplify a project's creation.

Essential Questions

- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How are algorithms implemented and executed on computers and computational devices?
- How are programs developed to help people, organizations or society solve problems?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- How do people develop and test computer programs?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- Slope Y-Intercept Project (to be developed)

Other:

- Python for Informatics: Exploring Information (http://www.pythonlearn.com/book.php)
- Runestone Interactive: How to Think Like a Computer Scientist (http://interactivepython.org/runestone/static/CCPS_Python/index.html)
- Turtle graphics API documentation for Python 3.4.1: https://docs.python.org/3.4/library/turtle.html (https://docs.python.org/3.4/library/turtle.html)
- Circle Lab (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01.html (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01.html))
- Triangle Lab (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01a.html (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01a.html))

Lesson Plan

Session 1

Getting Started (10 min)

- Display *Python for Informatics: Exploring Information* Chapter 4 (Functions) - Section 4.11.
- Students record in their journals four reasons to use functions and discuss with elbow partners to identify their two most important reasons.
- Ask students to identify two or three people they know who have the same name.
  - Explain to their elbow partners who the people are and how they tell them apart when talking about them.
  - Explain that we are going to working with functions today and that sometimes functions use the same names for different variables.
  - Before writing programs with functions, we want to address the possibility of functions that use the same names.

Exploring Functions (40 min)

Teaching Note:

Use of a format such as Google forms is sugggested for collecting student responses to the questions for each activity. It is important to keep this portion of the class moving so students have enough time for the labs. If students work with partners during their program development, they will have someone to share their progress with. Use a timer and have students briefly share their progress roughly every 10 minutes. This will not only help them understand an iterative development process, but also gives students practice with a collaborative development style.

Debugging

- Display Section 4.12 of *Python for Informatics*.
- Explain: when progam developers write their own functions, they often run into problems (bugs) they need to fix. These debugging suggestions should help the students to prevent or correct problems.
- As students investigate and work with functions today, they should an eye out for function design problems and strategies used to prevent or solve them. We will add the rules we develop to this list.

Variables and Parameters

- Students should open the Runestone Interactive Functions page (http://interactivepython.org/runestone/static/CCPS_Python/Functions/functions.html (http://interactivepython.org/runestone/static/CCPS_Python/Functions/functions.html)).
- Have students read "Variables and parameters are local" and run the Codelens to completion.
- Ask:
  - What caused the error that occurred when the last line of the code was executed?
  - What happens to local variables when a function finishes?
  - Are parameter's values that are assigned inside a function also accessible outside the function?

Global Variables (Don't Use Them!)

- Run ActiveCode 9.
- What is the value of the variable `power` inside the function?
- At what point was the variable `power` given that value?
- This form of communication is risky, since programmers may not realize that a variable used in one code segment is also used inside a function elsewhere in the program.
- What is the recommended way to communicate values to a variable?
- Emphasize that avoiding global variables supports collaboration and reuse of code, since different authors can avoid interfering with each other's work.

Variable Scope

- Complete the "Check for Understanding" after CodeLens 5, answering questions func-10, func-11, and func-12.
- Share "Check for Understanding" results either with elbow partners or the class as a whole.

Accumulating Values

- Go to the section entitled "The accumulator pattern."
- Do ActiveCode 10 and CodeLens 6.
- After reviewing the execution in the CodeLens, answer the "Check for Understanding" questions func-13 through func-14.
- See if the students can add any rules for creating and using functions to the list from *Python for Informatics*.

Functional Decomposition

- Go to the section entitled "Functions can call other functions".
- Do CodeLens 7 and ActiveCode 12.
- Ask:
  - What function is started first when the code in ActiveCode 12 is executed?
  - What function is finished first when the code is executed?

Session 2

Function Labs (45 min)

- Preview the three lab projects that students will complete and help students to get started with the Slope Y-Intercept project.
  - Discuss the iterative development process after watching the video "Design as an Iterative Process"
  - http://www.youtube.com/watch?v=9NxWW4poljU (http://www.youtube.com/watch?v=9NxWW4poljU)
- Complete the Slope Y-Intercept Project and the Drawing a Circle Labs.
  - The Slope Y-Intercept Project is described in the document of the same name.
  - The document includes descriptions of two functions to complete and contains a small sample of test data.
  - The Slope Y-Intercept Project is in the Lesson Resources folder along with the starter Python file.
- Continue in Runestone to the Lab entitled Drawing a Circle (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01.html (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01.html)).
  - In this second lab, students develop a function that uses a turtle to draw a circle.
  - After finishing the lab, students should copy their code to a Python module in PyCharm.
  - Students should create three functions named `drawSquare`, `drawTriangle`, and `drawOctagon`, along with main module code that calls each function.
- After completing the Drawing a Circle lab, students should follow the link below to the Lessons from a Triangle lab.
  - http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01a.html (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01a.html)
- Students should create a second Python module with the code from the starting point for the `drawPolygon` function code and finish implementing the `drawPolygon` function.
- Students should complete the Function Labs by developing the `drawCircle` function in Finally a Circle (the second half of the Lessons from a Triangle lab).
  - Additional components of the lab can be used as extensions for more able students.

Wrap-Up (5 minutes)

- After submitting the labs, students present their labs with elbow partners and groups.

- Students reflect on the following prompts about writing programs with functions.
- What problems did they encounter?
- How did they overcome the problems?
- How do functions simplify the task of developing a program?

Options for Differentiated Instruction

Iterative development works by developing, then sharing, programs at many points during the development process. Students should work in pairs as they create their programs and share the work through various completion stages.

Students who are completing projects quickly should be introduced to the Python turtle API at (https://docs.python.org/3.4/library/turtle.html). Students can investigate and implement such methods as `fill`, `speed` and others as described by the API documentation.

Evidence of Learning

Formative Assessment

Check for understanding by assessing student performance on the Runestone Interactive questions. Students should first try to resolve any difficulties with their partners and groups.

Students should be able to make suggestions for creating and using functions.

Identify and address any areas discovered that students have been unable to come to a consensus understanding.

Summative Assessment

Create functions that receive parameters, perform calculations using those parameters, and return a value.

Write a function to return the slope and y-intercept of a function of the line through two points.

Write functions to create a variety of polygons and a circle.

Internal Use Only

Pending Tasks

Unit 3 resources (Y-Intercept lab; anything else?) need to be made available when curriculum is published.

Future Work

Need to think about where students can get more practice with independent problem solving and program design. So far all programming activities have been heavily structured/guided.

(http://csmatters.org) 2 - 15

0b10 - 0b1111

Strings: Traversing, Slicing, and Parsing

Unit 2. Developing Programming Tools

Revision Date: Jul 21, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary

Students will use the online book *Python for Informatics* to complete a two-session guided lab in which they will explore the use of strings in Python.

Outcomes

Students will be able to:

- Identify a string as a sequence of characters that are identified by their index value, beginning with 0 (zero)
- Use the `len` function to get the number of characters in a string
- Traverse strings using both while and for loops
- Slice strings using `[m:n]`
- Parse strings using the `find` method and slicing
- Debug simple string programs to find and correct problems

Overview

Session 1:

1. Getting Started (5 min)
2. Guided Activity (45 min)
    1. Preparation [5 min]
    2. Code Review [40 min]

Session 2:

1. Getting Started (5 min)
2. Guided Activity (35 min)
    1. Preparation [5 min]
    2. Code Review [30 min]
3. Summative Assessment (10 min)

Learning Objectives

CSP Objectives

- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
    - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
- 4.1.1 - Develop an algorithm for implementation in a program.
    - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
    - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
    - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
    - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
    - 4.1.1H - Different algorithms can be developed to solve the same problem.
    - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
    - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
    - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
    - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
    - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
    - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
    - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.
    - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.2.1 - Explain how programs implement algorithms.
    - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
    - 5.2.1B - Program instructions are executed sequentially.
    - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
    - 5.2.1D - An understanding of instruction processing and program execution is useful for programming.
- 5.3.1 - Use abstraction to manage complexity in programs.
    - 5.3.1I - Strings and string operations, including concatenation and some form of substring, are common in many programs.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
    - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
    - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.

    - 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
    - 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
    - 5.4.1K - Correctness of a program depends on correctness of program components, including code segments and procedures.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
    - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
    - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
    - 5.5.1F - Compound expressions using and, or, and not are part of most programming languages.
    - 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.

Math Common Core Practice:

- MP7: Look for and make use of structure.
- MP8: Look for and express regularity in repeated reasoning.

Common Core ELA:

- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.10 - Read and comprehend science/technical texts

NGSS Practices:

- 5. Using mathematics and computational thinking

Key Concepts

- Strings are made up of individual characters in a sequence. The characters are identified by their position in the string; this position is referred to as an *index value*.
- Index values in Python begin with zero (0).
- Python provides a number of built-in functions and methods that allow a programmer to manipulate strings.
- Strings can be traversed using both `for` and `while` loops. Students must decide which loop structure is appropriate, based on the nature of a program's requirements.

Essential Questions

- How can computing and the use of computational tools foster creative expression?
- How are algorithms implemented and executed on computers and computational devices?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- *Python for Everybody (formerlt Python for Informatics)* by Charles Severance, http://www.pythonlearn.com/book.php (http://www.pythonlearn.com/book.php).
  This book is available for download as a .pdf file. We recommend downloading the book and making it available on a shared group drive on local computers for easy access for students. Is it licensed under Creative Commons and is free to download and share. A copy of the pdf is available in the Lesson Resources folder.

Other:

- More information on the `find` method for Python 3.4 can be found at: https://docs.python.org/3.4/library/stdtypes.html?highlight=find%20method#str.find (https://docs.python.org/3.4/library/stdtypes.html?highlight=find%20method#str.find)
- Multiple choice quiz (located in the Lesson Resources folder).

Lesson Plan

Session 1

Getting Started (5 min)

Journal Question: What is a string?

- Have students share their ideas.

Guided Activity (45 min)

Preparation [5 min]

- Students should open the book *Python for Informatics* by Charles Severance, Chapter 6: Strings (page 67). (Available at http://www.pythonlearn.com/book.php (http://www.pythonlearn.com/book.php) and in the lesson folder.)
- Students should launch PyCharm and make a new file called `stringtester.py` to test code as they go though the lesson.
- Remind students to place parentheses around the contents of the `print` function, as the book was written prior to Python 3.x.
- Recommend teachers project their screens with the book and PyCharm environment when appropriate.
- Have students test code samples from the book as the students and teacher go through the lesson together.

Code Review [40 min]

Section 6.1: *A string is a sequence*

- Introduce the concept of index values for each character in a string, beginning with 0 (zero).
- After reading the brief text in Section 6.1, have students type the sample code for fruit and letter with [ ] surrounding the index value of a letter and run it. Have students try to 'break' the code by typing in an index value too large for the word and see what kind of error they get.
- Code check for understanding: have students type in a different word for their fruit variable and change the index value. Does the printed letter match what was anticipated?
- Concept check for understanding: reinforce understanding by writing sample words on board, pointing to a letter at random and asking what the index value of that letter.
- Repeat checks for understanding with made-up problems as needed.

Section 6.2: *Getting the length of a string using len*

- Introduce the `len` function, which returns the number of characters in a string. Explain that empty spaces between words and punctuation also count as characters. Remind students of previous lessons concerning ASCII values for all characters, including blank spaces and punctuation.
- Using the `fruit` variable from the previous sample, have students return the number of characters using the `len` function.
- Stress the point in the book concerning `IndexErrors:` that the length of the string and the highest index value in that same string are not the same value. The length of the string `banana` is 6; the index values are numbered from 0 to 5, inclusive.
- Code check for understanding: Have students type in sample code that will return the length of a variety of strings, with and without spaces and punctuation. Did their code run without error?
- Concept check for understanding: Write a sentence on the board with spaces and punctuation. Have students count and write the "length" of the sentence on a post-it note and stick them on a designated place. Have a student or two organize the post-its based on the answers. Together with the class, count out the value of `len` , then have students type the sentence into the console in PyCharm to check their answers.
- Repeat checks for understanding with made-up problems as needed. Some famous quotes can be used to keep it interesting, such as, "The artist is nothing without the gift, but the gift is nothing without work." - Emile Zola (1840-1902).

Section 6.3: *Traversal through a string with a loop*

- Introduce the concept of printing each character of a string on a separate line by using a `while loop and incrementing the index value of each character.`
- Students should already be familiar with iteration using both while and for loops from previous lessons. Review syntax for both as needed.
- Have students traverse the string by running the sample code.

Session 2

Getting Started (5 min)

Journal Question: Explain why the length of a string is one digit higher than the highest index value of the same string.

- Have students share their answers. Guide the discussion back to the previous lesson, in which students traversed a string with a loop.
- Share a sample of code similar to the traversal with a while loop from the last lesson and ask students to determine the output based on an evaluation of the code. Example:

```
word = "alphabet"
index = 0
while index < len(word):
   letter = word[index]
   print (letter)
   index = index + 1
```

- Remind students of the shortcut to increment index, using `index += 1` in place of `index = index + 1` .

Guided Activity (35 min)

Preparation [5 min]

- Students should open the book *Python for Informatics* by Charles Severance, Chapter 6: Strings (page 67). (Available at http://www.pythonlearn.com/book.php (http://www.pythonlearn.com/book.php) and in Lesson Resources folder.)
- Students should launch PyCharm and open the `stringtester.py` file used to test the code that they created during the last lesson.

Code Review [30 min]

Section 6.3: *Traversal through a string with a loop continued*

- Take students through a discussion of traversing a loop backwards by beginning at the end of the length of the string minus 1, and decrementing the loop control variable (index) instead of incrementing.
- Code check for understanding: Book Exercise 6.1: Write a while loop that starts at the last character in the string and works its way backwards to the first character in the string, printing each letter on a separate line, except backwards.

Sample solution:

```
fruit = "watermelon"
length = len(fruit)
index = length - 1

while index >= 0:
   letter = fruit[index]
   print(letter)
   index -= 1
```

- Concept check for understanding: Project your code with errors and have students write the errors they find on post-it notes and stick them on a designated space. Have a student, or small group of students, organize the post-its in a way which makes sense to them. Debug the program together as a class based on the post-it note results.
- Introduce the concept of using a `for` loop instead of a `while` loop to traverse a string.
- Code check for understanding: Have students type sample code found at the top of page 69 in the book, using a `for` loop to traverse a string.
- Concept check for understanding: Ask students to think about when it is more helpful to use a `while` loop than a `for` loop to traverse a string and share their ideas. Expect answers such as “it is easier to make a `for` loop” and “it is harder to move backwards through a `for` loop than a `while` loop.”
- Explain that the word `char` in the example is used as a variable and is *not* a keyword in Python. It can be replaced with another word or letter. Have students use `x` instead of `char` in their sample `for` loop code to test this concept. This is an opportunity to have a discussion about using meaningful variable names so they have self-documenting code.

Section 6.4: *String slices*

- Students should return to page 69 in the book. Introduce the concept of slicing strings using the `[n:m]` syntax to indicate returning a string from the nth character to the mth character, not including the m^{th} character.
- Code checks for understanding: have students type in sample code which will slice strings in all the various ways covered in the section. For example:

```
print(fruit[0:3]) # returns app because a is in the 0th position and the second p is in the 2nd position.
                  # The slice goes to ‘p’ not including the ‘p’.

print(fruit[:3]) # Still returns app because a is in the 0th position.
                  # Leaving the first index blank begins the slice at the beginning of the string,
                  # and the second p is in the 2nd position.
                  # The slice goes to ‘p’ not including the ‘p’.
print(fruit[3:]) # Returns le because l is in the 3rd position.
                  # Leaving the second index blank ends the slice at the end of the string.
print(fruit[:]) # Returns apple because n begins at zero and m goes to the end of the string.
```

- Concept check for understanding: answer the question for Exercise 6.2: Given that `fruit` is a string, what does `fruit[:]` mean?
- Note: If you have time, you may want to go through Sections 6.5 - 6.9. Section 6.9 explains the difference between functions and methods, which will be useful when students use the find method while parsing strings.

Section 6.10: *Parsing strings*

- Introduce the `find` method and apply slicing in this section.
- As you go through this section, pay special attention to the `find` method. Explain that the `find` method returns the position at which the substring we are searching for begins.
- Code check for understanding: Have students type in practice code other than what is in the book and use the `find` method without error.
- Sample code:

```
   message = 'Meet me at the clock tower @ 7:00 a.m.'
   atSign = message.find('@')
   print (atSign)
```

This code sample returns `27`, indicating that the @ sign is at index value 27 in the string.

Summative Assessment (10 min)

- Coding assessment and/or quiz. Alternatively, the assessment may be assigned as homework.

Options for Differentiated Instruction

Students can work in pairs while new concepts are introduced and practiced.

One advanced student could be assigned to be the "checker" for each row and have them raise a flag or something similar when they have checked off everybody in their row as having one small group of programming exercises complete and correct. Possibly offer a token prize to the winning row.

For example, after Section 6.3, the following exercise could be assigned:

1. Everybody choose a different fruit.
2. Write code to display the first vowel in the name of the fruit and the first consonant.
3. Display the length of the fruit.
4. Print each character of their fruit name on a separate line by using a while loop.

Evidence of Learning

Formative Assessment

Code checks for understanding and concept checks for understanding are provided with each new function, method, or concept introduced.

Summative Assessment

Summative coding assessment:

- Assign your first and last name to a variable.
- Use the find method to locate the space between your first and last names.
- Use slicing to isolate and print only your first name.

Sample answer code:

```
name = 'Pat Miller'
space = name.find(' ')
firstName = name[0:space]
print (firstName)
```

It is recommend that a 10-question multiple choice quiz that requires students to evaluate code samples from these lessons, determining the output or possible outcomes when the code is run be developed.

Internal Use Only

Future Work

Perhaps instead on relying on the the online textbook for the lesson, provide examples for a teacher-led discussion, followed with the students completing the textbook examples on their own. This would rely less on the online textbook in case it changes or is unavailable.

(http://csmatters.org) 2 - 16

0b10 - 0b10000

Lists: Creation, Traversal, Insertion, and Removal

Unit 2. Developing Programming Tools

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary

This lesson answers the question "What is a list and what can I do with one?". Students will find the answer to this question by exploring list operations and methods, as well as investigating how lists are modeled in real-world situations.

Outcomes

- Students will create lists, access, and traverse elements within the list.
- Students will perform list operations including insertion, concatenation, repetition, slices, and deletion.
- Students will be introduced to list methods including append, insert, pop, sort, reverse, index, count, and remove.
- Students will explore the Python API.
- Students will use lists to model several real-world situations.

Overview

Session 1

1. Getting Started (5 min)
    1. Introduction [3 min]
    2. Discussion [2 min]
2. Guided Activities (40 min)
    1. Listy Linkys RolePlay [20 min]
    2. Python Documentation [10 min]
    3. List Slices [10 min]
3. Wrap Up (5 min)
    1. Homework

Session 2

1. Getting Started (5 min)
2. Guided Activities (40 min)
    1. Lists are Mutable [20 min]
    2. Paired Programming [20 min]
3. Wrap Up (5 min)
    1. Homework

Learning Objectives

CSP Objectives

- 1.2.4 - Collaborate in the creation of computational artifacts.
    - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
- 1.3.1 - Use computing tools and techniques for creative expression.
    - 1.3.1E - Computing enables creative exploration of both real and virtual phenomena.
- 4.1.1 - Develop an algorithm for implementation in a program.
    - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
    - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
    - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
    - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
    - 4.1.1E - Algorithms can be combined to make new algorithms.
    - 4.1.1H - Different algorithms can be developed to solve the same problem.
    - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
    - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.

    - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
    - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
    - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
    - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
    - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
    - 5.1.1C - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution.
    - 5.1.1D - Additional desired outcomes may be realized independently of the original purpose of the program.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
    - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
    - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
- 5.1.3 - Collaborate to develop a program.
    - 5.1.3A - Collaboration can decrease the size and complexity of tasks required of individual programmers.
    - 5.1.3B - Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.
    - 5.1.3C - Collaboration in the iterative development of a program requires different skills than developing a program alone.
    - 5.1.3D - Collaboration can make it easier to find and correct errors when developing programs.
    - 5.1.3E - Collaboration facilitates developing program components independently.
    - 5.1.3F - Effective communication between participants is required for successful collaboration when developing programs.
- 5.2.1 - Explain how programs implement algorithms.
    - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
    - 5.2.1B - Program instructions are executed sequentially.
    - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
    - 5.2.1D - An understanding of instruction processing and program execution is useful for programming.
    - 5.2.1E - Program execution automates processes.
    - 5.2.1J - Simple algorithms can solve a large set of problems when automated.
- 5.3.1 - Use abstraction to manage complexity in programs.
    - 5.3.1H - Data abstraction provides a means of separating behavior from implementation.
    - 5.3.1I - Strings and string operations, including concatenation and some form of substring, are common in many programs.
    - 5.3.1K - Lists and list operations, such as add, remove, and search, are common in many programs.
    - 5.3.1L - Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.
    - 5.3.1M - Application program interfaces (APIs) and libraries simplify complex programming tasks.
    - 5.3.1N - Documentation for an API/library is an important aspect of programming.
    - 5.3.1O - APIs connect software components, allowing them to communicate.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1A - Program style can affect the determination of program correctness.
    - 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
    - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
    - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.
    - 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
    - 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
    - 5.4.1L - An explanation of a program helps people understand the functionality and purpose of it.
    - 5.4.1M - The functionality of a program is often described by how a user interacts with it.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
    - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
    - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
    - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
    - 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.
    - 5.5.1H - Computational methods may use lists and collections to solve problems.
    - 5.5.1J - Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP4: Model with mathematics.
- MP7: Look for and make use of structure.

Common Core Math:

- F-IF.1-3: Understand the concept of a function and use function notation
- F-IF.4-6: Interpret functions that arise in applications in terms of the context

Common Core ELA:

- WHST 12.5 - Develop and strengthen writing as needed by planning, revising, editing, rewriting
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products

NGSS Practices:

- 2. Developing and using models

Essential Questions

- How can computing and the use of computational tools foster creative expression?
- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- "Warm-Up Get Ready for Lists"
- "LinkyListy Role Play"

Other:

- Python API for Built-in Types (https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range) - https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range (https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range)
- http://interactivepython.org/runestone/static/CCPS_Python/Lists/lists.html (http://interactivepython.org/runestone/static/CCPS_Python/Lists/lists.html)
- Original collegeboard horse barn question #3: http://apcentral.collegeboard.com/apc/public/repository/ap_frq_computerscience_12.pdf (http://apcentral.collegeboard.com/apc/public/repository/ap_frq_computerscience_12.pdf)

Lesson Plan

Session 1

Getting Started (5 min)

Introduction [3 min]

Review the string commands. As they enter the class, hand students a copy of the Warm-Up Get Ready for Lists.docx that is found in the Lesson Resources folder.

Discussion [2 min]

Discuss: What is a list? (A sequential collection of Python data values; each value is identified by an index.)

Guided Actvities (40 min)

List Role Play [20 min]

See the file calld LinkyListy Role Play in the Lesson Resources folder. Note: You can change student names to students within your class.

- Create a List of students by starting with an empty list and then appending students. Have each student come up to stand in front of the room where memory resides. Give each student her/his index on a notecard as s/he is appended to the list. Ask students which index number they think that they should get. What does this numbering system remind you of? It is just like strings! For cxample: `students = []; students.append("Mary"); students.append( “Jack”); students.append("Jill")` . Note: If we enclose the elements of the list in square brackets when appending multiple items, they are added as a sublist, not as individual elements.
  - Length of a `list: print(len(students))`
  - Accessing Elements: `print(students[2])`
  - List Membership (in, not in): `print(“Mary” in students)`

- Concatenation and repetition (+ *):Make a list of more Python commands on sentence strips. On back of the sentence strip, write the output or trace of the statement after it has executed onto the back of the card. Have one student hold the deck of cards and cycle through them. Ask students to read a card and predict the output. Meanwhile, students who are in the list will change positions to demonstrate the behavior within the list. The holder of the card provides feedback or congratulatoins in checking the correctness of classmate's responses.
- Include cards that model these behaviors:

```
more = ["Tom", "Laverne"]
students = students + more
pets =['fish'*3,'dog']*2          # creates a list of ['fishfishfish','dog','fishfishfish','dog']
```

- List deletion using the delete operator: `del students[2], del students[2:4]`
- List deletion using the List method pop: `stu = students.pop(2)` , `stuLast = students.pop()` . The pop method pops (deletes and returns) an element at a given index or the last element if no index is provided.
- Extension: additional list methods that may be useful and interesting: sort, reverse, index, count.

Check for understanding: Ask students to give an example and then explain the effect of several of the List methods. For example, `students.append("Zoe")` would add Zoe to the end of the list of students. `students.insert(4, "Larry")` would add Larry at index position 4 and slide everyone else down one slot. This could be done as a placemat activity. Use different colored markers for each student to write the example. Turn placement and ask the next student to explain the effect.

Python Documentation [10 min]

- Check out the Python docs for Built-in Types.
- Go to Section 4.6 Sequence types- list, tuple, range, and check out the sections on (4.6.1) Common Sequence Operations, (4.6.3) Mutable Sequence Types, and (4.6.4) Lists.
- Students need to realize that Application Programmer Interfaces (APIs) are an important resource for programming.
- Ask several questions regarding the info found in the API. (Link to resource: Python API for Built-in Types (https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range))

List Slices [10 min]

- Quickly resurrect your list of students.
- This time, write names on the board.
- To simulate list slices, add each student's age and hair color after each student's name.
- Yes, lists can contain different kinds of objects!
- Ask students to copy the list and write the code to pull out a sublist that contains the second student's name and info.

Wrap Up (5 min)

Discussion: How are lists the same and how are they different from Strings?

Homework

Have the students do the Runestone lab activities for Lists to reinforce the above concepts.
(http://interactivepython.org/runestone/static/CCPS_Python/Lists/lists.html
(http://interactivepython.org/runestone/static/CCPS_Python/Lists/lists.html))

Session 2

Getting Started (5 min)

- Create a list that contains the integers from 1 to 10.
- Create the same list from an empty list using a loop.
- Check for understanding: create a list that counts down from 10 to 1 and then adds "Blastoff".
- Reverse a given list.

Guided Activities (40 min)

Lists are Mutable [20 min]

Demonstrate that lists are mutable with the following activity

- Create a list representing a horse barn containing 4 horses in 6 stalls.
- Use the built-in constant `None` for the empty stalls.
- Find an empty stall.
- If there is an empty stall, place "Toby" in the Barn in that slot.
- Also. see if a particular horse is in the barn or if he is out to pasture.
- Exchange 2 horses in the barn. (Adapted from 2012 APCS exam question 3, page 13 of http://apcentral.collegeboard.com/apc/public/repository/ap_frq_computerscience_12.pdf (http://apcentral.collegeboard.com/apc/public/repository/ap_frq_computerscience_12.pdf))

Paired Programming [20 min]

- In a group of 2-3, make a zoo of animals.
- Demonstrate the use of at least 6 different list operations and methods.
- For a bonus, try to make a story with your code.

Wrap Up (5 min)

Homework

Create a list of 5 students that contain the students' name, age, and hair color. Use a loop to extract the information for each student and print it out.

Teacher Note: Consider using the YumYumCupcake problem (see Formative Assessment) as part of tomorrow's opening exercises.

Options for Differentiated Instruction

- Use a canvas shoe bag to demonstrate a list.
- Use a parking lot with numbered spaces as an example of a list.

Evidence of Learning

Formative Assessment

- Create a quiz similiar to the role play. Use the answer chart provided as a template for the quiz.
- Check for understanding: Ask students to explain what the effect of the List methods append, insert, etc. For example, students.append “Zoe” would add Zoe to the end of the list of students? Students.insert(4, “Larry”). Would add Larry at index position 4 and slide everyone else down one slot?
- Check for understanding: Create a list that counts down from 10 to 1 and then adds “Blastoff”.
- Let's visit the YumYum Shoppe: Create a list that has different kinds of cupcakes and simulate the actions that might take place during a typical day. For example, `yumYumCupcakes = ["chocolate mousse" *3, "vanilla creme", "strawberry fluff", "chocolate mousse"*2].` Have a customer purchase a vanilla creme cupcake if there are any, check how many chocolate mouse cupcakes are in the display case, bake some more, and add them in. Drop one cupcake on the floor and throw it away.

Summative Assessment

Paired programming: Make a zoo of animals and demonstrate the use of at least 6 different list operations and methods. Try to make a story with your code.

Internal Use Only

Pending Tasks

The Day Two pair programming activity ("make a zoo") seems underspecified and might be hard for teachers and students to interpret. Developing a clearer in-class assignment here, and maybe a homework assignment to let the students reinforce what they've learned, would be useful.

(http://csmatters.org) 2 - 17

0b10 - 0b10001

Unit Assessment

Unit 2. Developing Programming Tools

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 3 50-minute sessions

Lesson Summary

Summary

To conclude the unit, students will complete a small project as well as a written assessment. The project requires students to parse text and search through lists or words to find a specific characteristic. The assessment covers integers, strings, booleans, loops, if statements, and lists.

Outcomes

- Students will synthesize concepts from the previous lessons to create their first project in PyCharm.
- Students will design a function and determine the relationship between algorithms and functions.

Overview

1. Getting Started (5 min)
2. Independent Activity (40 min)
3. Wrap Up (5 min)

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
  - 1.2.3A - Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
  - 1.2.3C - Combining or modifying existing artifacts can show personal expression of ideas.
- 1.2.4 - Collaborate in the creation of computational artifacts.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
  - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
  - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
  - 4.1.1E - Algorithms can be combined to make new algorithms.
  - 4.1.1F - Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.
  - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
  - 4.1.1H - Different algorithms can be developed to solve the same problem.
  - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
  - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
  - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
  - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
  - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
  - 4.1.2I - Clarity and readability are important considerations when expressing an algorithm in a language.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
  - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
  - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
  - 5.1.1D - Additional desired outcomes may be realized independently of the original purpose of the program.
- 5.1.2 - Develop a correct program to solve problems.
  - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
  - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
  - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
  - 5.1.2D - Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.
  - 5.1.2E - Documentation about program components, such as code segments and procedures, helps in developing and maintaining programs.
  - 5.1.2F - Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments.
  - 5.1.2G - Program development includes identifying programmer and user concerns that affect the solution to problems.
  - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.

    - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.2.1 - Explain how programs implement algorithms.
    - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
    - 5.2.1B - Program instructions are executed sequentially.
    - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
    - 5.2.1D - An understanding of instruction processing and program execution is useful for programming.
    - 5.2.1I - Executable programs increase the scale of problems that can be addressed.
    - 5.2.1J - Simple algorithms can solve a large set of problems when automated.
- 5.3.1 - Use abstraction to manage complexity in programs.
    - 5.3.1A - Procedures are reusable programming abstractions.
    - 5.3.1B - A procedure is a named grouping of programming instructions.
    - 5.3.1C - Procedures reduce the complexity of writing and maintaining programs.
    - 5.3.1D - Procedures have names and may have parameters and return values.
    - 5.3.1E - Parameterization can generalize a specific solution.
    - 5.3.1F - Parameters generalize a solution by allowing a procedure to be used instead of duplicated code.
    - 5.3.1G - Parameters provide different values as input to procedures when they are called in a program.
    - 5.3.1H - Data abstraction provides a means of separating behavior from implementation.
    - 5.3.1I - Strings and string operations, including concatenation and some form of substring, are common in many programs.
    - 5.3.1J - Integers and floating-point numbers are used in programs without requiring understanding of how they are implemented.
    - 5.3.1K - Lists and list operations, such as add, remove, and search, are common in many programs.
    - 5.3.1L - Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
    - 5.4.1D - Longer code segments are harder to reason about than shorter code segments in a program.
    - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
    - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.
    - 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
    - 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
    - 5.4.1I - Programmers justify and explain a program's correctness.
    - 5.4.1J - Justification can include a written explanation about how a program meets its specifications.
    - 5.4.1K - Correctness of a program depends on correctness of program components, including code segments and procedures.
    - 5.4.1L - An explanation of a program helps people understand the functionality and purpose of it.
    - 5.4.1M - The functionality of a program is often described by how a user interacts with it.
    - 5.4.1N - The functionality of a program is best described at a high level by what the program does, not at the lower level of how the program statements work to accomplish this.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
    - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
    - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
    - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
    - 5.5.1F - Compound expressions using and, or, and not are part of most programming languages.
    - 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.
    - 5.5.1H - Computational methods may use lists and collections to solve problems.
    - 5.5.1J - Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP5: Use appropriate tools strategically.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.
- MP8: Look for and express regularity in repeated reasoning.

Common Core Math:

- S-ID.1-4: Summarize, represent, and interpret data on a single count or measurement variable

NGSS Practices:

- 3. Planning and carrying out investigations
- 5. Using mathematics and computational thinking
- 8. Obtaining, evaluation, and communicating information

Key Concepts

Students should synthesize concepts from the previous lessons to create their first project in PyCharm. This lesson also pushes a student to think about how to design a function, and the relationship between algorithms and functions.

Essential Questions

- How are algorithms implemented and executed on computers and computational devices?
- What kinds of problems are easy, what kinds are difficult, and what kinds are impossible to solve algorithmically?
- How are algorithms evaluated?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

What are the key elements we need to think about when designing a function?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- Word Play
- Word Play Rubric
- Assessment

Lesson Plan

Getting Started (5 min)

- Ask the students to work in small groups to create pseudocode for the following function:
- Create a function titled `'is_palindrome'` that inputs a word and determines whether that word is a palindrome. If it is, return `True`, otherwise return `False`.
- As a class, use the groups' pseudocode to create the function `'is_palindrome'`.
- Introduce the projects to students. They will have a total of 2 sessions to complete their project.

Independent Activity (40 min)

Students work individually on the Word Play and Assessment which are found in the lesson resource folder.

Wrap up (5 min)

Allow students to continue working to the end of class on their projects; have individual check-ins with students to make sure that they are on track and have a clear idea of what they need to complete the following day.

Options for Differentiated Instruction

Option to allow students to complete Word Play with partners to promote collaboration, then complete the written assessment individually.

Evidence of Learning

Formative Assessment

Teacher will monitor the progress of the students on each of the programs in Word Play.

Summative Assessment

Written assessment (see google drive)

Project Assessment (see google drive for project and rubric)

Internal Use Only

Future Work

Will create key for the final assessment.

(http://csmatters.org) 3 - 1

0b11 - 0b1

The Internet: Basics of Information Transmission

Unit 3. Information and the Internet

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

In this lesson, students will analyze what the Internet is and its basic functionality. Students will learn how the Internet works and how the implementation of the Internet has affected our society. They will discuss the idea of the Internet as a delivery service to get bits from one place to another.

Outcomes

- Students will explain the characteristics of the Internet and how the systems built on it influence their use.
- Students will explain the difference between bandwidth and latency.
- Students will analyze relationships of data transfer over the systems within the Internet.
- Students will synthesize how data transfer and Internet systems are affected by the environment and needs of its users.

Overview

1. Getting Started (5 min) - Students discuss the difference between the Internet and a browser.
2. Introduction (15 min) - Students explore how the Internet has grown over time and read from the "Blown to Bits" book.
3. Guided Activity (13 min) - The path that the Internet uses to send and receive information is explored and diagrammed.
4. Demonstration (5 min) - The teacher presents the concept of bandwidth using different websites.
5. Guided Activity (10 min) - The class discusses the uses of real-time Internet usage.
6. Independent Activity (Optional)
7. Wrap Up (2 min) - Students each present what they learned in the lesson.

Learning Objectives

CSP Objectives

- 6.2.1 - Explain characteristics of the Internet and the systems built on it.
  - 6.2.1A - The Internet and the systems built on it are hierarchical and redundant.
  - 6.2.1C - IP addresses are hierarchical.
  - 6.2.1D - Routing on the Internet is fault tolerant and redundant.
- 6.2.2 - Explain how the characteristics of the Internet influence the systems built on it.

    - 6.2.2I - The size and speed of systems affect their use.
    - 6.2.2J - The bandwidth of a system is a measure of bit rate - the amount of data (measured in bits) that can be sent in a fixed amount of time.
    - 6.2.2K - The latency of a system is the time elapsed between the transmission and the receipt of a request.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.

7.1.1 MNO

Math Common Core Practice:

- MP3: Construct viable arguments and critique the reasoning of others.
- MP5: Use appropriate tools strategically.
- MP7: Look for and make use of structure.

Common Core ELA:

- RST 12.2 - Determine central ideas and conclusions in the text
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.6 - Analyze the author's purpose in providing an explanation, describing a procedure
- RST 12.7 - Integrate and evaluate multiple sources of information presented in diverse formats and media
- RST 12.8 - Evaluate the hypotheses, data, analysis, and conclusions in a science or technical text
- RST 12.9 - Synthesize information from a range of sources
- RST 12.10 - Read and comprehend science/technical texts
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products
- WHST 12.7 - Conduct short as well as more sustained research projects to answer a question
- WHST 12.8 - Gather relevant information from multiple authoritative print and digital sources, using advanced searches effectively; assess the strengths and limitations of each source
- WHST 12.9 - Draw evidence from informational texts to support analysis, reflection, and research

NGSS Practices:

- 1. Asking questions (for science) and defining problems (for engineering)

NGSS Content:

- HS-ETS1-1. Analyze a major global challenge to specify qualitative and quantitative criteria and constraints for solutions that account for societal needs and wants.

Key Concepts

The Internet and the systems built on it have a profound impact on society.

Essential Questions

- What is the Internet, how is it built, and how does it function?
- What aspects of the Internet's design and development have helped it scale and flourish?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

Teacher Resources

Student computer usage for this lesson is: **required**

Blown to Bits (Abelson, Ledeen, Lewis): http://www.bitsbook.com/ (http://www.bitsbook.com/)

Optional Student Handout: Internet Change Student Handout
Answer Key for Teacher: Internet Change Answer Key

The following links provide background on the topics covered in this lesson:

- http://www.policy.hu/inetclass/arpaNet.html (http://www.policy.hu/inetclass/arpaNet.html)
- https://www.youtube.com/watch?v=JUs7iG1mNjI (https://www.youtube.com/watch?v=JUs7iG1mNjI)
- http://googlemapsmania.blogspot.com/2013/12/mapping-internet-of-things.html (http://googlemapsmania.blogspot.com/2013/12/mapping-internet-of-things.html)
- http://www.google.com/green/storyofsend/desktop/ (http://www.google.com/green/storyofsend/desktop/) (defunct as of 10/1/2014)
- http://www.webperformancetoday.com/2012/04/02/latency-101-what-is-latency-and-why-is-it-such-a-bigdeal/ (http://www.webperformancetoday.com/2012/04/02/latency-101-what-is-latency-and-why-is-it-such-a-bigdeal/)
- http://www.coolnerds.com/Newbies/Bandwidth/Bandwidth.htm (http://www.coolnerds.com/Newbies/Bandwidth/Bandwidth.htm)

- http://www.cnet.com/internet-speed-test/ (http://www.cnet.com/internet-speed-test/)
- http://www.speedtest.net/ (http://www.speedtest.net/)
- http://www.bandwidthplace.com/ (http://www.bandwidthplace.com/)
- http://www.akamai.com/html/technology/dataviz1.htm (http://www.akamai.com/html/technology/dataviz1.html)l

Lesson Plan

Getting Started (5 min) - Discussion: What is the Internet?

Start with a VERY brief class discussion: How does the Internet work?

Journal: what is the difference between the Internet and a browser? How are they connected?

- Internet: A collection of systems working together to deliver data to the user. This includes email, video streaming, social media, websites, cloud storage like DropBox, etc.
- Browser: Software applicaton designed to retrieve and display information from the world wide web, which is one part of the Internet, for the user.

Introduction (15 min)

Part 1 - Activity (5 min)

(Use the optional Student Handout for Unit 3 Lesson 1 if desired to guide all of the activities in this lesson. There is an answer key to the student handout for teachers to use as well.)

Demonstrate, using the following resources, how the Internet has grown from the small ARPANET system to what it is today.

1. ARPANET image: http://www.policy.hu/inetclass/arpaNet.html (http://www.policy.hu/inetclass/arpaNet.html) (from 1971)
2. *Today Show* discussion on "What is the Internet?" (2:12 min):https://www.youtube.com/watch?v=95-yZ-31j9A (https://www.youtube.com/watch?v=95-yZ-31j9A)
3. Image representing today's Internet: http://googlemapsmania.blogspot.com/2013/12/mapping-internet-of-things.html (http://googlemapsmania.blogspot.com/2013/12/mapping-internet-of-things.html) (Note: zoom in to see the details, investigate your local area you can see your own school, experiment, this is fun, have students find 4 different things of 4 different colors)

Part 2 - Reading (10 min)

Students read the following sections from the "Blown to Bits" book (Online book link: http://www.bitsbook.com/wp-content/uploads/2008/12/B2B_3.pdf (http://www.bitsbook.com/wp-content/uploads/2008/12/B2B_3.pdf)):

- "The Internet as a Communication System"
- "Packet Switching"
- "Core and Edge"

These sections are on pages 301-303 in the pdf version.

Guided Activity (13 min)

Part 1 - Diagram (3-5 min)

Students create a diagram of how an email might travel from its start point to the end point.

Part 2 - Discussion (7-10 min)

1. Discuss the flow of an email as a group. (Use this tool to supplement the understanding of the process: *https://www.youtube.com/watch?v=5Be2YnlRIg8 (https://www.youtube.com/watch?v=5Be2YnlRIg8)* stop at about 50 seconds.
2. Explain that packets are limited by "bandwidth" and "latency." The teacher may have the students read the following articles (both are relatively short) and lead a discussion after, or use them as resources for their own presentation. Measured in bits per second = how many bits arrive.
    - Latency: http://www.webperformancetoday.com/2012/04/02/latency-101-what-is-latency-and-why-is-it-such-a-big-deal/ (http://www.webperformancetoday.com/2012/04/02/latency-101-what-is-latency-and-why-is-it-such-a-big-deal/)
        - Latency is the delay. How long you have to wait to send or receive.
    - Bandwidth: http://www.coolnerds.com/Newbies/Bandwidth/Bandwidth.htm (http://www.coolnerds.com/Newbies/Bandwidth/Bandwidth.htm)
        - Bandwidth is like the width of a highway, the more lanes you have the more cars can drive on the road at once

Demonstration (5 min)

Demonstrate how to test bandwidth using the following sites:

- http://www.cnet.com/internet-speed-test/ (http://www.cnet.com/internet-speed-test/)

- http://www.speedtest.net/ (http://www.speedtest.net/)
- http://www.bandwidthplace.com/ (http://www.bandwidthplace.com/)

Guided Activity (10 min)

1. Use the Internet Usages Guide in the Lesson Resources folder to lead a discussion of real-time internet usage
2. Look at types of questions that could be answered based on this tool: http://www.akamai.com/html/technology/nui/industry/ (http://www.akamai.com/html/technology/nui/industry/)

Independent Activity (Optional)

Students should use the tool to come up with a question that can be answered by the tool. Students write an analysis that asks the question, answers the question and provide proof of why the answer is correct by providing screenshots of the tool in their report. (http://www.akamai.com/html/technology/nui/industry/ (http://www.akamai.com/html/technology/nui/industry/)) or http://www.internetlivestats.com/ (http://www.internetlivestats.com/) for live Internet stats.

(Note: Students can possibly start this assignment in class, but will likely need to complete as homework.)

Wrap-up (2 min)

In this activity, students will each share one thing they have learned from this lesson. This can be done in several ways depending on time constraints or disabilities. All students should participate in some way before leaving the classroom.

- Have all students stand. In order to sit back down, students must share one thing they have learned to the class. (This may happen organically, or in a prescribed order.)
- If there is a disabled student for whom the standup/sit activity would not be an option, develop an alternative way to indicate who in the class has answered (such as having them raise their hands).
- If there is a student who has difficulty speaking in front of the class or there isn't sufficient time, hand out index cards on which each student must write what they have learned, to be handed in as an exit ticket to leave the class.

Options for Differentiated Instruction

- Students can share comparisons of assignments in small groups.
- Students can further explore net usage (using http://www.akamai.com/html/technology/dataviz1.html (http://www.akamai.com/html/technology/dataviz1.html)) specifically targeting mobile usage and/or broadband usage by geographical regions. Analyze the differences between these geographical regions of packet usage.

Evidence of Learning

Formative Assessment

Using a real-time network tool that measures the number of views per minute, students generate a question that can be answered using this tool. They will then collect the data and write a report that answers this question. The report should use current real-time screenshots for data and examples. (Note: Students can possibly start this assignment in class, but will likely need to complete as homework.)

Summative Assessment

Possible question(s) to use for a future test:

- What is the relationship between bandwidth and latency?
- When using Internet tools to display data, what are some important factors that need to be considered to better understand the information being displayed?

Internal Use Only

Future Work

This seems like a long lesson - it might be worthwhile to change some of the activities to homework.

Consider the following resource: http://www.akamai.com/html/technology/nui/industry/ (http://www.akamai.com/html/technology/nui/industry/)

(http://csmatters.org) 3 - 2

0b11 - 0b10

The Internet: Present and Future

Unit 3. Information and the Internet

Revision Date: Jul 18, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

The Internet is growing to connect to everything we do in our lives. Over the years, it has grown from being a representation of static content, to web 2.0: a place where users interact to a collection of users and "things." In this lesson, the students will conceptualize devices that collect data and send it through the Internet.

Outcomes

- Students will understand the development of the Internet.
- Students will understand how devices communicate on the Internet.
- Students will imagine/design things (that don't yet exist) that could connect to the Internet.

Overview

1. Getting Started (10 min) - Journal and discussion on devices that use the Internet.
2. Guided Activities (35 min) - Students explore the "Internet of Things" through videos and readings.
3. Wrap Up (5 min) - The teacher leads a discussion on class comprehension of the topic.

Learning Objectives

CSP Objectives

- 6.1.1 - Explain the abstractions in the Internet and how the Internet functions.
  - 6.1.1A - The Internet connects devices and networks all over the world.
  - 6.1.1B - An end-to-end architecture facilitates connecting new devices and networks on the Internet.
  - 6.1.1C - Devices and networks that make up the Internet are connected and communicate using addresses and protocols.
  - 6.1.1D - The Internet and the systems built on it facilitate collaboration.
- 6.2.1 - Explain characteristics of the Internet and the systems built on it.
  - 6.2.1A - The Internet and the systems built on it are hierarchical and redundant.
- 6.2.2 - Explain how the characteristics of the Internet influence the systems built on it.
  - 6.2.2D - Interfaces and protocols enable widespread use of the Internet.
  - 6.2.2I - The size and speed of systems affect their use.
  - 6.2.2J - The bandwidth of a system is a measure of bit rate - the amount of data (measured in bits) that can be sent in a fixed amount of time.
  - 6.2.2K - The latency of a system is the time elapsed between the transmission and the receipt of a request.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
  - 7.1.1M - The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.
  - 7.1.1O - The Internet and the Web have impacted productivity, positively and negatively, in many areas.
- 7.2.1 - Explain how computing has impacted innovations in other fields.
  - 7.2.1C - Computing enables innovation by providing the ability to access and share information.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
  - 7.3.1K - People can have instant access to vast amounts of information online; accessing this information can enable the collection of both individual and aggregate data that can be used and collected.

7.1.1 MO

7.2.1 C

7.3.1 K

Common Core ELA:

- RST 12.7 - Integrate and evaluate multiple sources of information presented in diverse formats and media
- WHST 12.1 - Write arguments on discipline specific content
- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products
- WHST 12.9 - Draw evidence from informational texts to support analysis, reflection, and research

Key Concepts

The Internet is an ever-evolving system of increasing complexity. It has evolved from representing static information to providing interactivity of data between users and objects (things).

Essential Questions

- How can computational models and simulations help generate new understanding and knowledge?
- What is the Internet, how is it built, and how does it function?
- What aspects of the Internet's design and development have helped it scale and flourish?
- How does computing enhance human communication, interaction, and cognition?
- How does computing enable innovation?
- What are some potential beneficial and harmful effects of computing?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

Teacher Resources

Student computer usage for this lesson is: **optional**

Students need access to paper for documentation.

Blown to Bits (Abelson, Ledeen, Lewis). Text is free as pdf: http://www.bitsbook.com/ (http://www.bitsbook.com/)

Access to Internet connectivty for these links/videos:

- https://www.youtube.com/watch?v=fFqEx--b7hU (https://www.youtube.com/watch?v=fFqEx--b7hU)
- http://www.zdnet.com/the-internet-of-things-outlook-for-2014-everything-connected-and-communicating-7000024930/ (http://www.zdnet.com/the-internet-of-things-outlook-for-2014-everything-connected-and-communicating-7000024930/)
- http://www.ibm.com/smarterplanet/us/en/overview/article/iot_video.html (http://www.ibm.com/smarterplanet/us/en/overview/article/iot_video.html)

In the Lesson Resources Folder:

- "Commercial Python Project" Project Description Document
- "Commercial Python Project Rubric" Rubric for the Commercial Project

Lesson Plan

Getting Started (10 min) - Journal / Discussion

1. In their journals, ask students to identify as many objects in the room as they can that are connected to the Internet (or that would be more useful if they were connected to the Internet).
2. Have students share with a neighbor. Then, communicate through a whip-around or a large group discussion. Generate a list of devices.
3. Review investigations of Internet usage statistics that the students completed in Lesson 3-1 (particularly useful if the students completed the investigation as homework).

Guided Activities (35 min)

The next three activities are used to generate ideas for examples of "things" that either are connected or could be connected to the Internet.

Part 1 - Video (5 min)

Show the video (no audio except music) on how an average everyday person uses objects connected to the Internet in our current society: https://www.youtube.com/watch9v=fFqEx--b7hU (https://www.youtube.com/watch?v=fFqEx--b7hU) (3:58)

Summary: A day in the life of the Internet of things shows these things connected to the Internet: cell phone, thermostat in the house, car entry system and radio, car GPS intelligently looking for available parking, parking sensors on the ground using mesh networking (short-range connections to a larger deployment system in a central box), a heart rate monitor with results that can be viewed online in real time, a watch that connects with a cash register/inventory system, a package pickup system that connects with a drone to take the package directly to the customer.

Part 2 - Reading (5 min)

- Before reading the article, ask students to find the ONE term used in the beginning of the article that many teens would not know (possible answer: ubiquitous)
- Ask students to read the article: http://www.zdnet.com/the-internet-of-things-outlook-for-2014-everything-connected-and-communicating-7000024930/ (http://www.zdnet.com/the-internet-of-things-outlook-for-2014-everything-connected-and-communicating-7000024930/) (~2.5 pages)
    - See CSP_Unit2_Lesson2_StudentHandout and AnswerKey in the lesson folder for guided questions for students to answer and a teacher's answer key.
    - OR a simpler assignment would be to ask students to read only the introduction of this article (stop at "Big Data"), and answer these 3 questions:
        1. Who coined the term "Internet of Things" and WHEN? **Kevin Ashton in 1999**
        2. What was the very first "Internet of Things" -- the first "thing" connected to the Internet? (Hint -- not the refrigerator!) **The Coke Machine at Carnegie-Mellon University's Computer Science department**
        3. What are three "far-reaching" implications of the "Internet of Things"? **(answers will vary)**

Part 3 - Video (5 min)

- Show on of these videos about how data is generated by devices connected online : http://www.ibm.com/smarterplanet/us/en/overview/article/iot_video.html (http://www.ibm.com/smarterplanet/us/en/overview/article/iot_video.html) (5:25)
  Summary: Tons of data has been generated; now it's instrumented and documented. Billions of people and things are using the Internet: traffic sensors, flow rate monitors, more things on the Internet than people. It's a sea of data. DIKW triangle = data, information, knowledge, wisdom. When you apply intelligence, it transforms from one form to another (data into information, information into knowledge). The ideal day: your laptop knows your schedule so it knows when to wake you, chooses best transportation, preheats bathroom and warms up car, tailored communication on what's happening around you that affects you. If the parts could cooperate, they could make smart decisions about utility usage and other decisions. There are sensors everywhere--underneath your feet, in taxis, trains, and buses. If they communicate, then a serious water blockage could change traffic patterns to allow police to arrive quickly before a disaster strikes.
- https://www.youtube.com/watch?v=uEsKZGOxNKw (https://www.youtube.com/watch?v=uEsKZGOxNKw) (2.25)
  Summary: visual display of how data is collected and visualized.

Part 4 - Discussion (5 min)

Analyze with students in discussion what objects they saw in the previous videos and readings that they use. Were there any objects that they did not think about that are connected to the Internet? Adjust the list as needed.

Small Group Activity (15 min)

With a partner, imagine a device that might someday be a part of the "Internet of Things," but currently does not exist. An example might be a shoe that has its own wireless acquired IP address and keeps track of how many steps one takes each day. (**Note**: This may already exist.)

As a small group, the students should submit a document answering the following questions:

- What is the purpose of this device?
- What data will your device collect?
- What sensors will it use?
- Who will make use of the data?
- What will be the range of values needed to store the data?

This document should also include a sketch of the device.

Wrap Up (5 min)

Students display a thumbs up or thumbs down to this question: Did this lesson help you comprehend the concept of the Internet as an entity that is comprised of both people (users) and objects or machines?

Homework

Read *Blown to Bits* (Pg 303 - 306) - IP Addresses - stop at "The Key to It All: Passing Packets."

*Optional Project for Additional Python Practice ** Note: highly recommended!*

The document in the Lesson Resources folder called "Commercial Python Project" is a project designed to give students Python coding practice and allow them to explore more about the "Internet of Things" by creating their own product commercial template. Consider adding the requirement that their program include conditional statements. Extra time will be needed.

The rubric for this project can also be found in the Lesson Resources Folder

Options for Differentiated Instruction

Ask students to think about and document how their selected device may have an impact on our daily lives. Could their be any controversy associated with their device or the use of their device? If so, what is that controversy? Students should document their opinions and/or findings.

When selecting the pairs, aim for diversity of background, so the students learn how others view technology.

Evidence of Learning

Formative Assessment

With a partner, imagine a device that might someday be part of the Internet of Things, but currently does not exist.

As a group, the students should submit a document answering the following questions:

- What is the purpose of this device?
- What data will your device collect?
- What sensors will it use?
- Who will make use of the data?
- What will be the range of values needed to store the data?

This document should also include a sketch of the device.

Summative Assessment

How does the Internet effectively connect devices and networks?

How do devices and networks that make up the Internet communicate?

Internal Use Only

Future Work

- create a "reading guide" - something for students to write (short questions) as they read the Blown to Bits homework assignment

- Nora created a worksheet with questions related to the reading (on Piazza, 10/1/14)

(http://csmatters.org) 3 - 3

0b11 - 0b11

How the Internet Works: Routing

Unit 3. Information and the Internet

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

This lesson delves deeper into the structure of the Internet and routing protocols. Students will explore the necessity of redundancy by using packets to transmit sections of data. They will then discuss standards for packets and routing. The class will simulate a network in which each student is a node through which they will send email packages from one node to another.

Outcomes

- Students will explain how the Internet moves data from one place to another using routers.
- Students will understand how data is encoded on the Internet.
- Students will explain how large amounts of data are managed on the Internet.

Overview

1. Getting Started (10 min)
2. Activity Pt A: Simulation of packet transfer (10 min)
3. Activity Pt B: Simulation including lost packets (10 min)
4. Discussion (15 min)
5. Wrap Up (5 min)

Learning Objectives

CSP Objectives

- 2.3.1 - Use models and simulations to represent phenomena.
  - 2.3.1A - Models and simulations are simplified representations of more complex objects or phenomena.
- 3.3.1 - Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information.
- 6.1.1 - Explain the abstractions in the Internet and how the Internet functions.
  - 6.1.1C - Devices and networks that make up the Internet are connected and communicate using addresses and protocols.
  - 6.1.1E - Connecting new devices to the Internet is enabled by assignment of an Internet protocol (IP) address.
- 6.2.1 - Explain characteristics of the Internet and the systems built on it.
  - 6.2.1D - Routing on the Internet is fault tolerant and redundant.
- 6.2.2 - Explain how the characteristics of the Internet influence the systems built on it.
  - 6.2.2A - Hierarchy and redundancy help systems scale.
  - 6.2.2B - The redundancy of routing (i.e., more than one way to route data) between two points on the Internet increases the reliability of the Internet and helps it scale to more devices and more people.
  - 6.2.2F - The Internet is a packet-switched system through which digital data is sent by breaking the data into blocks of bits called packets, which contain both the data being transmitted and control information for routing the data.
  - 6.2.2G - Standards for packets and routing include transmission control protocol/Internet protocol (TCP/IP).
  - 6.2.2H - Standards for sharing information and communicating between browsers and servers on the Web include HTTP and secure sockets layer/transport layer security (SSL/TLS).
- 6.3.1 - Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it.
  - 6.3.1B - The DNS was not designed to be completely secure.

Math Common Core Practice:

- MP3: Construct viable arguments and critique the reasoning of others.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.

NGSS Practices:

- 1. Asking questions (for science) and defining problems (for engineering)
- 2. Developing and using models
- 3. Planning and carrying out investigations
- 8. Obtaining, evaluation, and communicating information

Key Concepts

Students will be able to:

- ***Create*** a list of aspects of the Internet's design that have helped it scale and flourish, and ***articulate*** how these aspects contribute to its growth.
  Diagram the path of an email as it travels from one Internet user to another.
- ***Explain*** why it makes sense to send data in multiple packets rather than all together.

Essential Questions

- What is the Internet, how is it built, and how does it function?
- What aspects of the Internet's design and development have helped it scale and flourish?
- How is redundancy built into the Internet?

Teacher Resources

Student computer usage for this lesson is: **none**

1. Materials required

- Post-It Notes - each student needs 10-20 Post-Its
- Desks/Tables should be moved to form a grid such as a 6 by 6 grid.

2. Copies to make

- Teacher IP Address Locations - One for the teacher.
- Student IP worksheet - One per student. Either hand these out after students are seated, or put one at each desk prior to the start of class. Make sure that the layout of the worksheets matches the layout of the Teacher IP Address Locations.

3. Digital resources (check for access)

- https://www.youtube.com/watch?v=WwyJGzZmBe8 (https://www.google.com/url?q=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DWwyJGzZmBe8)

4. Required background knowledge

- http://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm (http://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm)
  - Particularly the IP and Routing Hierarchy sections.
- http://www.dummies.com/how-to/content/exploring-tcpip-routers.html (http://www.dummies.com/how-to/content/exploring-tcpip-routers.html)
  - Routers
- https://www.youtube.com/watch?v=RbY8Hb6abbg (https://www.youtube.com/watch?v=RbY8Hb6abbg)
  - How IP works (This can be shown to students at the discretion of the teacher.)

Lesson Plan

Getting Started (10 min)

Journal

Prompt students to respond in their journals to one or more of these questions:

- When you type the name of a website (URL) into your browser, how does the browser know how to find that website?
- How does data get from one point to another on the Internet?

Discussion: Invite students to share their journal entries. The class should come to the general consensus that while their computer doesn't know where to find everything on the Internet, it is able to pass information or requests from one location to another.

- As an analogy, describe the theory behind "six degrees of separation." This theory states that if person A is told the name of one other person in the world (person B), then through no more than five individuals, one of whom is a personal acquaintance, person A will be able to contact person B. The choice of that personal acquaintance is key: If you're trying to find someone in France, it is better to ask a friend in France to find person B than to ask your neighbor (unless you already know that your neighbor has strong connections with the people in France near person B). While the theory isn't precisely true, it is true that most real-world networks tend to have "hubs" (highly connected nodes (people who know many other people, or Internet nodes that are connected to many other nodes) and "spokes" (connections from hubs to individuals or nodes who are less well connected).

Activity Part A: Simulation of Packet Transfer (10 min)

Transition Remark: Previously, we looked at the general structure of the Internet and how it works. Today, we will look more closely at the process of sending information between two locations using the Internet. Let's see what this looks like through a World of Science video (https://www.youtube.com/watch?v=WwyJGzZmBe8). (After video) We are going to simulate this same action by sending packets of information to each other without leaving our seats.

Introduction:

- Tell students that they will use rules, just like the Internet, called *protocols*. Compare the process followed by the Internet to get information from one place to another to the algorithms that they created previously (inputs of some type are given to the algorithm; it performs the same process on any information given; and then it produces a result). The protocols are applied to every packet of information that is sent. For this simulation, our protocol will have the structure "Recipient IP:____, part # of total, Sender IP:_____ "
- On the Internet, the addresses are called Internet Protocol (IP) addresses. These addresses are made up of three numbers, and often correlate to your geographical location. Today, you will each be given an IP address consisting of three letters. I have determined this address through your geographical location in the room. (As students participate in this activity, they will see the use of hierarchy. Most of their packets that begin with the same letter will go to the same general region, although there are a few that break this pattern.)

- Your goal is to send a letter that I will hand out to the correct IP address, but here is the catch: Computers do not store the location of every IP address in the world. They are given the IP addresses of other coputers that are connected to them. Thus, you may only communicate with your neighbors to tell them your IP address. This must be done silently. (You may show them your IP address from your paper, or write it on a Post-It.)
- You can also ask your neighbors (via Post-It note) who they have access to, and your neighbors can ask the same question of their neighbors. In this way, you may find that you have access to someone else, but not know the route that the package must go to get there.
- On the back of your IP card, you have a table, which you should use to record how to get messages between yourself and different students in the class.
    - **Example:** We are going to attempt to transmit a package from B.B.A to C.B.D.
        - Student B.B.A - ask neighbors if they are C.B.D, or can get there.
        - This request should propagate through students until someone has found C.B.D.
        - C.B.D responds to their neighbor, who tells the neighbor who asked. This repetition should continue until the news has reached the original sender. B.B.A gives their message to the neighbor with the connection, and records on their paper who they went through to make the connection.
        - The message is passed on to C.B.D, who opens it, and reads it.
- Hand each student paper to serve as "packets," and allow them to send messages to one another. Make sure that they use the correct protocol on each message.
- Allow students to send packets that request information (what is your favorite ice cream, do you have siblings, etc.), and make sure that return packets are addressed and sent.

Activity Part B (10 min)

Transition Remark: Our simulation of the protocol system on the Internet has been relatively tame. In reality, it doesn't always work this nicely. Sometimes packets are lost; not all the information you want to transmit fits in one packet; or some routers are unable to keep working. Fortunately, the Internet is full of redundancy that allows it to keep working even if some parts fail to work, and we can send large data sets through multiple packets. We're going to run our simulation again, but this time living in the "real world."

- Allow students to send messages to one another, but this time, as students transmit packets, mix them up a bit to simulate lost packets or unreadable data.
- Require students to find information from one another, but give them a character limit (like a text or tweet) for each packet that requires them to use multiple requests to send the information.
- Give IP cards some identifiable characteristic (print them on different color paper, put a sticker in the corner, etc.), and tell students with a particular characteristic that they are unable to connect to the network. Ask the remainder of the class to try to send messages without them. This will simulate the power of redundancy.

Discussion: How does redundancy of routers contribute to Internet fault tolerance?

- Some of your packets are getting lost! Sometimes this occurs in the middle of a multi-packet message. This is very frustrating, especially when we do not know whether all of our packets were received. How could we make our protocol system better in order for our Internet to run more smoothly?
    - This discussion should end with students deciding it would be nice to have a reply message such as "I got it!" with the number the packets that went together (1 of 3). The teacher should allow students to come to this conclusion on their own, but you may need to push them in this direction.
- Return to simulation, test your new protocols. (Spend no longer than 5 minutes.)

Discussion (10 min)

Transition Remark: We just participated in a simulation that allowed us to become nodes within the Internet. By filling out the back side of your IP worksheet, each of you was essentially becoming a *router*. Each routers contains a *configuration table* with information that it can use to send packets to the correct location.

Discussion:

- What is the role of the IP address for each device on the Internet?
- How does redundancy of routers contribute to the Internet's ability to scale to more connections?
    - Students will need to extrapolate from their understanding of the simulation to answer this question.

Conclusion: This information should be written in the student's journal.

- We have simulated some protocols used to transfer information across the Internet. However, there are other protocols that are necessary for sharing information and communicating between browsers and servers on the Web. These include information exchange (HTTP protocols) and secure sockets layer/transportation layer security (SSL/TLS).

Wrap Up (5 min)

Reflection: This may be completed as an exit ticket for formative assessment or in student journals.

- In the activity, what happened when you tried to send out packets of information?
- What worked in the activity and what did not work?
- Make comparisons between what happened in the activity and what actually happens as data moves on the Internet.

Options for Differentiated Instruction

Routers

- Research how routers use IP addresses to know where to send packets of information. Write one paragraph, a poem or rap, or create a diagram to communicate what you learned on the topic.

Traceroute

- Use a Traceroute utility to trace the path that a packet takes to get to your computer. Create a flowchart/path diagram to record what you learned.

Code.org Lesson

- For an alternative lesson or to reinforce concepts, use Unit 2 Lesson 4: "Routers and Redundancy" from Code.org. https://docs.google.com/document/d/1-m0yDvgTkM10N6N9-WKdFE2wd2BrMIiHL6hmTSxy-Zs/edit (https://docs.google.com/document/d/1-m0yDvgTkM10N6N9-WKdFE2wd2BrMIiHL6hmTSxy-Zs/edit)

Evidence of Learning

Formative Assessment

Assessment will occur informally through the discussion questions:

- What is the role of the IP address for each device on the Internet?
- How does redundancy of routers contribute to the Internet's ability to scale to more connections?

Reflection questions for journal:

- In the activity, what happened when you tried to send out packets of information?
- What worked in the activity and what did not?
- Make comparisons between what happened in the activity and what actually happens as data moves on the Internet.

Summative Assessment

Assessment Questions:

Explain the role of the Internet Protocol address for each device on the Internet.

Why is the assignment of an IP address critical to connecting a device to the Internet?

How does Internet router redundancy contribute to Internet fault tolerance?

How does redundancy of routers contribute to the Internet's ability to scale to more connections?

Explain how relatively small packets are used to transmit large files on the Internet and identify what information each packet must possess.

Identify a standard protocol for Internet packet communication.

Internal Use Only

Pending Tasks

Put final versions of worksheets online in folder.

Future Work

I'd really like to see a new teacher work through this and see what needs further explanation

(http://csmatters.org) 3 - 4

0b11 - 0b100

How the Internet Works: Domain Name System

Unit 3. Information and the Internet

Revision Date: Aug 21, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Pre-lesson Preparation

Students must complete the pre-reading assignment: Blown to Bits (pages 303 - 306 in the PDF). They should read the sections about DNS, Protocols, and IP Addresses.

Summary

The purpose of the Domain Name System is to resolve domain names to IP address for computers on the Internet.

For the next two lessons, students will investigate the workings of the Domain Name System (DNS). They will then design and enact a simulation of DNS. Students will use their simulation to request and receive web pages, implement DNS caching, and investigate DNS poisoning.

Outcomes

- Explain the abstractions in the Internet and how DNS supports Internet functions.
- Explain the hierarchical characteristics of DNS.
- Explain how IPv4 addressing is used to identify and connect computers on the Internet.

Overview

Session 1 - Introduce DNS

1. Lesson Introduction (5 min) - Introduce DNS.
2. Guided Activity (40 min) - Students investigate elements of DNS and sketch its components.
3. Closing (5 min) - Watch improvisation video.

Session 2 - Create DNS Improvisation

1. Introduce Simulation/Improvisation (5 min).
2. Guided Activity (40 min) - The roles and scripts for the next lesson's activity are set up.
3. Closing (5 min) - Journaling about the characteristics of DNS and the Internet.

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
  - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4C - Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation.
  - 1.2.4E - Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.
- 2.3.1 - Use models and simulations to represent phenomena.
- 6.1.1 - Explain the abstractions in the Internet and how the Internet functions.
  - 6.1.1B - An end-to-end architecture facilitates connecting new devices and networks on the Internet.
  - 6.1.1E - Connecting new devices to the Internet is enabled by assignment of an Internet protocol (IP) address.
  - 6.1.1F - The Internet is built on evolving standards, including those for addresses and names.
  - 6.1.1G - The domain name system (DNS) translates domain names to IP addresses.
- 6.2.1 - Explain characteristics of the Internet and the systems built on it.
  - 6.2.1B - The domain name syntax is hierarchical.
- 6.2.2 - Explain how the characteristics of the Internet influence the systems built on it.

    - 6.2.2C - Hierarchy in the DNS helps that system scale.
- 6.3.1 - Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it.
    - 6.3.1B - The DNS was not designed to be completely secure.

Math Common Core Practice:

- MP5: Use appropriate tools strategically.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.

Common Core ELA:

- RST 12.2 - Determine central ideas and conclusions in the text
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.10 - Read and comprehend science/technical texts
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products
- WHST 12.9 - Draw evidence from informational texts to support analysis, reflection, and research

NGSS Practices:

- 1. Asking questions (for science) and defining problems (for engineering)
- 2. Developing and using models
- 3. Planning and carrying out investigations

Key Concepts

Characteristics of the Internet influence the systems built on it.

Students should be able to explain how computers can be used to get a web page from a new web server.

Essential Questions

- What is the Internet, how is it built, and how does it function?
- What aspects of the Internet's design and development have helped it scale and flourish?

Teacher Resources

Student computer usage for this lesson is: **required**

Blown to Bits (either electronic or hard copy)

- http://www.bitsbook.com/excerpts/ (http://www.bitsbook.com/excerpts/)

Access to the Internet for these sites:

- https://www.youtube.com/watch?v=72snZctFFtA (https://www.youtube.com/watch?v=72snZctFFtA)
- https://www.site24x7.com/find-ip-address-of-web-site.html (https://www.site24x7.com/find-ip-address-of-web-site.html)
- http://computer.howstuffworks.com/dns2.htm (http://computer.howstuffworks.com/dns2.htm)

Excel or similar software

"DNSWorksheet" document and "Favorite Domains (Sample List)" spreadsheet in lesson resources folder

Addtional Resources

https://studio.code.org/s/netsim (https://studio.code.org/s/netsim) Code Studio Internet Simulator

Student friendly explanation of DNS https://studio.code.org/s/netsim (https://studio.code.org/s/netsim)

Lesson Plan

Session 1 - DNS Introduction

Getting Started (5 min) - Introduction of DNS

1. Students will watch this video https://www.youtube.com/watch?v=72snZctFFtA (https://www.youtube.com/watch?v=72snZctFFtA) from 0:47 to 3:22 to introduce the function of DNS.

2. Students should consider the video and the pre-lesson reading to answer the following question: "What is the purpose of the Domain Name System?"
3. After the students have finished journaling, the teacher should give these explanations to clarify DNS and further set up the lesson:
    - The purpose of DNS is to resolve domain names to IP address for computers on the Internet.
    - Our purpose as a class is to understand what DNS means, how it works, what benefits it provides, and some challenges it faces.

Guided Activity (40 min)

Part 1 (30 min) - Investigation

Version 1 - If students have access to the system console window, use it to complete the following steps.

Directions for Host Configuration and DNS in Action Using the Console Window:

1. Console Based Use: (cmd prompt)
2. Find your host's IP address: (ipconfig)
3. Find the IP address of the default gateway (ipconfig)
4. Find the physical address of the default gateway (arp –a)
5. Resolve domain names: (nslookup)
    - google.com
    - umbc.edu
    - domain.name
6. Domain names recently resolved: View the HOST DNS cache (ipconfig /displaydns)
7. Unknown domain names: Find the IP address of the local DNS server: (ipconfig /all)

Version 2 - If the Console is blocked for your students, you can still demonstrate most of the console commands on your computer. If it is blocked for you as well, use a web site such as pingtool.org and the prompts below.

Note: If the Console is blocked, students will need a way to obtain unique IP addresses. A document named "DHCP Simulator" (in the lesson folder) contains 30 unique IP formatted addresses. Print and cut out the blocks and allow a student dubbed DHCP to give them out at random.

Directions for Host Configuration and DNS in Action Activity Without the Console Window:

1. Find your host's IP address: (just search for get ip address). Survey student results and explain that the values are the often the same because the network uses one computer to make connection outside the local network.
2. Watch the first 40 seconds of DNS Cache Poisoning (https://www.youtube.com/watch?v=1d1tUefYn4U (https://www.youtube.com/watch?v=1d1tUefYn4U))
3. Use traceroute to find the number of routers (hops) involved in getting to apcsprinciples.org.
4. Resolve domain names: (nslookup)
    - google.com
    - umbc.edu
    - domain.name
5. Domain names already resolved: Copy the IP addresses to a browser's web address bar and visit two of the domains from number 4.
6. Unknown domain names: Review the traceroute results to each domain. What is the IP address of the default gateway?

During the investigation, students should answer the following questions in their journal:

Journal Questions for Version 1:

Students should attempt to answer these questions based on the previous activity:

1. How do you access a command prompt?
2. What console command do we use to find our IP address and the IP address of the default gateway?
3. How can we find the physical address of the default gateway?
4. How do Internet bound packets get to the default gateway?
5. How can we look up (resolve) unknown IP addresses from the console?
6. How can we see a list of the domain names and IP address already resolved by our host computer?
7. How does your computer resolve unknown domain names into their addresses?

Suggested Answers

1. In Windows XP, click Start>run then type cmd.
2. At a console prompt, type ipconfig.
3. At a console prompt, arp -a.
4. IP packets are sent inside physically addressed frames.
5. Use nslookup followed by the domain name.
6. At a console prompt, ipconfig /displaydns
7. The computer obtains the IP address by requesting it from the local DNS server.

Journal Questions for Version 2:

Students should attempt to answer these questions based on the previous activity:

1. The Console is just one tool that can be used to find a computer's actual IP address. How is the Console accessed on your computer at home, if you have one?
2. Why does pingtool.org not give us our real IP address?
3. What is the role of the local DNS server?
4. What is the purpose of the routers on the Internet?
5. Why did domain.name not resolve to an IP address?
6. What do we call the list of domain names and IP addresses already resolved by our host computer?
7. How does a computer resolve unknown domain names into their addresses?

Part 2 (5 min) - Video

1. Students will watch this video https://www.youtube.com/watch?v=72snZctFFtA (https://www.youtube.com/watch?v=72snZctFFtA) to show the function of DNS.
2. Give students a copy of the DNS worksheet.

Closing Activity (5 min) - Journal

Have students pick one or more of the following questions to answer in their journals:

- Just as the original Internet developers did, we are using a trust model - assuming everyone is cooperating. Why do you think we used a trust model?
- The Internet's end-to-end architecture makes connecting to it simple. Why do you think an end-to-end architecture encourages invention and innovation of Internet-based systems?

Optional Activities:

Diagram (10 min)

Put the following list of DNS and other devices on the board. As a class, students are to create a diagram of the way devices 1 – 5 interact to resolve domain names (similar to the last picture in the overview (http://www.edline.net/pages/North_Point_High_School/Classes/1415_14-15-P79060-001/Textbooks/7502160004216717474/DNS_Overview).) Have students draw the picture on the board and agree that it is correct before they write it in their journals.

1. Host
2. Local DNS
3. Root
4. Top Level Domain
5. Authoritative Name
6. *Web Server*
7. *Router*

All the world is a stage, and all the men and women merely players (55 minutes)

Getting Started/Introducing Activity (5 min)

The teacher will explain the following activity to the students:

1. We are going to simulate the function of DNS as used to obtain web pages on the Internet, creating a sort of improvisational play.
2. Just as the original developers did, we will use a trust model:
    - Assume everyone is cooperating.
    - Use an "end-to-end architecture" that keeps the center of our network as simple as possible.
    - **(Have students record both of these characteristics of the Internet in their journals.)**
3. Together with DNS, web servers, and Internet routers, students playing the role of each device will request, resolve, and deliver web pages.
4. Along the way, we will likely run into several problems and failures (learning opportunities). As we do, we will resolve them together and record our insights in our journals.

Show DNS explained (https://www.youtube.com/watch?v=72snZctFFtA (https://www.youtube.com/watch?v=72snZctFFtA)) 3:20 - 5:39.

Guided Activity (40 min) - Improvisation Activity

Part 1 (30 min) - DNS Device Roles

1. Have students watch "The Way of Improvisation" https://www.youtube.com/watch?v=MUO-pWJ0riQ (https://www.youtube.com/watch?v=MUO-pWJ0riQ)
2. In small “author” groups, have students prepare scripts for students who will play a device.
    - Each group creates a card or page for their device.
        - One side of the page contains the device name and an explanation of its role. (The explanation is to be brief – fewer than 25 words but enough to give a non-expert the general idea of the role of the device.)
        - The other side of the paper contains a more detailed explanation of the service the device provides, including the devices with which it communicates, a list of the data required for the server to function, and an explanation of how the data is initially acquired.
3. Each group should print one draft copy of their scripts. Students share (rotate) their device scripts with another group.
4. Each group makes constructive criticisms and returns scripts to their authors. (If time permits, you may want to repeat this round.)
5. "Author" groups make revisions and print 4 copies of each device page. (Use front and back of the paper if possible.)
6. Collect these and keep them sorted by device.

Part 2 (10 min) - Web Servers

Teachers will explain that the Internet is much bigger than the Web, but for our simulation purposes, we will only be trying to access web pages from web servers.

1. Depending on the size of the class, pick two or three domain names to simulate, such as .com and .net. Select two domains from each top-level domain. (For instance, for .com, you might select google.com and amazon.com.)
2. Divide the class into groups such as the "google group", the "amazon group", and the other subdomains selected.
3. Each student in each group finds a unique page from their second-level domain and prints two copies of the first page of the web page each student selects.
4. Have students write a one-word title for their page on the top of the paper and submit it to you, placing the pages in piles by subdomain.

Options for Differentiated Instruction

When pairing up students in "Think - Pair - Share," use a random generator such as random.org (use list tool) to randomly pair students.

Evidence of Learning

Formative Assessment

Students create a list of things in their lives that are identified by unique numbers.

Summative Assessment

1. End to End Architecture 6.1.1B
A. Describe the “end to end” architecture of the Internet.
B. Explain how the “end to end” architecture facilitates connection of new devices.

2. Internet Names and Address Rules 6.1.1 E
A. Describe how computers are uniquely identified and connected on the Internet.

3. DNS Function 6.1.1 G
A. Briefly explain the primary use of the Domain Name System made by users of the Internet.

4. DNS Hierarchy 6.2.1 B
A. Describe the hierarchy of the Domain Name System.

(http://csmatters.org) 3 - 5

0b11 - 0b101

Optional

How the Internet Works: DNS Activity

Unit 3. Information and the Internet

Revision Date: Jul 15, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Pre-lesson Preparation

This lesson will require some room setup or prep for best delivery of instruction. Some of the setup should have been done in the previous lesson.

Summary

In this lesson, students will expand their knowledge of how DNS works by acting out a simulation of DNS in action and using it to retrieve web pages. This is a two-session lesson. The first session is for students to get the simulation functioning, with the teacher serving as director. As students realize they need to "fix" their implementation of the simulation (modify their scripts), they record the insights in their journals.

In Session Two, students take on different roles and conduct a dress rehearsal that is entirely student-led. Teachers then introduce DNS caching and DNS poisoning. Once the simulation is functioning, students will address both increased efficiency due to DNS caching, and cybersecurity concerns associated with DNS.

Outcomes

- Students will explore how the characteristics of the Internet influence the systems built on the Internet.
- Students will understand that Domain Name Servers (DNS) are essentially the "address book" of the Internet and store information to help Internet systems route requests and replies.
- Students will be able to explain how DNS hierarchy supports scaling on the Internet.
- Students will identify existing DNS cybersecurity concerns and potential options to address these issues.

Overview

Session 1 - Acting the Simulation

1. Lesson Introduction (5 min) - Students assign the cast and collect necessary data.
2. Guided Activity (40 min) - The teacher directs the first rehearsal of the play and introduces changes in IP and DNS.
3. Closing (5 min) - Think-Pair-Share

Session 2 - Round 2 of Simulations

1. Dress Rehearsal with Improvisation (15 min) - Practice Play
2. Rehearsal with Video (30 min) - Perform Play and Discuss DNS Caching and Poisoning
3. Closing (5 min) - Summary Report

Learning Objectives

CSP Objectives

- 2.3.1 - Use models and simulations to represent phenomena.
- 6.1.1 - Explain the abstractions in the Internet and how the Internet functions.
  - 6.1.1B - An end-to-end architecture facilitates connecting new devices and networks on the Internet.
  - 6.1.1E - Connecting new devices to the Internet is enabled by assignment of an Internet protocol (IP) address.
  - 6.1.1F - The Internet is built on evolving standards, including those for addresses and names.
  - 6.1.1G - The domain name system (DNS) translates domain names to IP addresses.
- 6.2.1 - Explain characteristics of the Internet and the systems built on it.
  - 6.2.1B - The domain name syntax is hierarchical.
- 6.2.2 - Explain how the characteristics of the Internet influence the systems built on it.
  - 6.2.2C - Hierarchy in the DNS helps that system scale.
- 6.3.1 - Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it.

- 6.3.1B - The DNS was not designed to be completely secure.

Math Common Core Practice:

- MP5: Use appropriate tools strategically.
- MP7: Look for and make use of structure.

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products
- WHST 12.7 - Conduct short as well as more sustained research projects to answer a question
- WHST 12.9 - Draw evidence from informational texts to support analysis, reflection, and research

Key Concepts

The characteristics of the Internet influence the systems built on it.

Domain Name Servers are essentially the "address book" of the Internet and store information to help Internet systems route transmission requests and replies.

A list of character protocols is provided as a resource. These may help students learn their roles.

Essential Questions

- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- What is the Internet, how is it built, and how does it function?
- What aspects of the Internet's design and development have helped it scale and flourish?

Teacher Resources

Student computer usage for this lesson is: **required**

This lesson requires extensive preparation.

Acquire:

24 envelopes per class – one or two per host per rehearsal and production.

Post-it Notes

One color for students to use to self-select roles.

One color for students to use to record DNS information.

One color for students to use to initiate requests.

Print:

One copy of character protocols for each student.

One copy of Routing Table.docx for each student router.

One copy of DNS cache for each root, TLD, ANS and local DNS server and each host.

Four copies of each device/character role page (web, router, root, TLD, ANS and local DNS and host).

Three or four copies of the first page web page, grouped by domain.

Lesson Plan

Session 1 - First Rehearsal (Part 3 of DNS Section)

Getting Started (5 min)

Warm Up:

Distribute post-it notes to each student.

- Students find their IP addresses and write their name and IP address on the post it note.
- Display the DNS device list on the board and use it to review the process by which DNS resolves domain names.

Casting Characters:

Beside each device listed on the board, there should be the number of students needed to play each role. As soon as students complete their post it notes, have them choose their role by placing the post it notes next to the device name.

Below are suggested numbers of actors per role for two class sizes. Students take their seats and add their name and IP address to their router table.

Device List	16 Students	30 students
Host	4	8
Local DNS	2	4
Root	1	1
Top Level Domain	2	3
Authoritative Name	4	6
Web Server	4	6
Router	1	2

Once students select a role, each device group should meet briefly to discuss what information they have to collect from the post-it notes on the board. They will go and obtain either a script that informs them what to do during the play, (how their device works) or all the printed web pages from their server.

Gathering Data:

Post this list of directions and allow devices to go to the board and obtain the required IP address information.

Before the play can start, these seven sets of data still have to be collected.

1. Web servers (students) need to "advertise" their web pages by making a list on the board of the web pages (one word per page) they have to offer.
2. Hosts and DNS servers complete a routing table for their table and give the routing table to their router.
3. Local DNS servers need to share their IP address with hosts.
4. Top-level domain servers need to share their domain names and IP addresses with the root server.
5. Authoritative name servers need to share their domain names and IP addresses with the top-level domain servers.
6. Web servers need to share their IP addresses with their authoritative name servers.
7. Routers need to complete the routing table for their group using the routing table (Routing Table.docx) provided.

Guided Activity (40 min) - Guided Rehearsal

Part 1 (30 min) - Rehearsal 1

Notes:

- During Round 1, the director can stop the action, provide direction, and restart the action.
- Actors should make notes in their journals of any stage directions, and make any changes or corrections to their scripts/role sheets as needed.

Steps to complete the play:

1. Select one Host to start. Hosts:
    a. Select a web page to request.
    b. To get the IP address of the web server, the HOST writes the domain of the page requested on a post it note, placing the request in an envelope, **does not seal it**, addresses the outside of the envelope (both from and to IP addresses), and sends the envelope via the Internet router.
2. Routers:
    a. Verify the address are correctly formatted and forward the envelope using their routing tables.
    b. Return envelopes not addressed properly – during dress rehearsal.
3. When the root server gets the request, it:
    a. Opens the envelope.
    b. Reads the top-level domain.
    c. Writes the IP address of the proper TLD server on the post-it note.

    d. Uses the return address on the envelope to send it back to the local DNS server.
4. The local DNS server:
    a. Opens the envelope.
    b. Uses the IP address to readdress the question to the appropriate TLD.
5. The TLD server:
    a. Repeats the process; writing the IP address of the proper ANS on the post it note.
    b. Addresses the envelope back to the local DNS server.
6. The local DNS server:
    a. Opens the envelope.
    b. Uses the IP address to readdress the question to the appropriate ANS.
7. The ANS:
    a. Writes the IP address of the desired web server on the post it note and circles it.
    b. Sends the envelope back to the local DNS server.
8. The local DNS server:
    a. Opens the envelope.
    b. Upon finding a circled IP address, it sends the envelope back to the HOST.
9. The HOST:
    a. Opens the envelope and replaces the post-it note with a request containing (only) the name of the web page desired.
    b. Addresses it using the circled IP address.
10. The web server:
    a. Receives the envelope.
    b. Opens it and replaces the post-it note with a printed version of the requested page.
    c. Readdresses the envelope to the HOST.
    d. Sends the page back.
11. Simulation completed! Celebrate when the requested web page arrives. Have everyone take a bow.

Part 2 (10 min) - Changes in DNS

Explain: Both domain names rules name and IP address rules have changed over time.

Have students watch these two videos:

- Watch the video regarding one important change regarding domain names: https://www.youtube.com/watch?v=1kFcxf8KAjg (https://www.youtube.com/watch?v=1kFcxf8KAjg) [0:55]
- Watch the video regarding IPv4 and IPv6. https://www.youtube.com/watch?v=-Uwjt32NvVA (https://www.youtube.com/watch?v=-Uwjt32NvVA) [2:20]

Afterwards, they should record responses to these two prompts:

- How are naming rules changing and why?
- How are IP address numbers changing and why?

Closing (5 min) - Think-Pair-Share

1. Before Round 2 (the dress rehearsal), have students make entries in their journals of any lessons learned. Have them share these lessons first with elbow partners, and then in groups.
2. Collect all scripts and web pages.
3. Present the DNS lesson summary project (DNS Summary).

Session 2 – Dress Rehearsal

Dress Rehearsal with Improvisation (15 min)

Set the stage:

Distribute scripts, envelopes, and post-it notes.

- The rehearsal process repeats from the previous session; however, students are to select a different role and to work out on their own the simulation/improvisation.

    Note: Use the character protocols from the previous session during the dress rehearsal, especially if the students are struggling.

- Select a host at random to start by requesting an available web page. A little later, cue a second host. If the system is working, cue a third host. Go slower if needed.
- Return the web pages to the web servers. Before going live, give students a chance to ask any questions and to record any observations in their brain books.

Rehearsal with Video (30 min)

Part 1 (20 min) - Opening Night (If the show is ready)

This play is improvisational with Hosts requesting whichever pages they want. Restart the system with all Hosts online.

1. Video the full DNS-based Web system in action.
2. Make a second video of the system acting slowly with only one Host. In the second video, have students narrate their actions.
3. Introduce the use of DNS caching.
4. If time permits, add the use of DNS caching to the system. (Have students request a number of pages from the same domain, so pertinent cached values accumulate quickly. If time is short, brainstorm benefits students anticipate for DNS caching.)
5. Have students record the benefits of DNS caching.

Part 2 (10 min) - Attack

Introduce the security problems associated with plain text messages and with DNS poisoning:

- Have the students watch https://www.youtube.com/watch?v=1d1tUefYn4U (https://www.youtube.com/watch?v=1d1tUefYn4U) [start at 0:52].
- Have students record potential problems of DNS, including the lack of encryption and the DNS poisoning of DNS haches.

Students will research this topic and report on security issues (among other aspects of DNS) in their summary report.

Closing (5 min)

If the previous lesson has not been assessed, assign the entire summary in the document entitled "DNS Summary."

Options for Differentiated Instruction

The simulation will be acted out at least three times. Students should change to a new role each time.

Use the character protocols during the dress rehearsal.

Evidence of Learning

Formative Assessment

Students are to reopen their spreadsheets from the previous lesson.

Show the students how to create a simple "if statement" in a spreadsheet. The spreadsheet will become a tool where a user can type a domain name into a cell. If the respective IP address of that domain name is found, then that IP address is shown. If it is not then a "0" appears.

(See the sample spreadsheet in the Lesson Resources folder called "Sample Spreadsheet")

Summative Assessment

A. Describe one rule change for names used on the Internet. Describe the rule before and after the change.

B. Describe one rule change for IP addresses used on the Internet. Describe the rule before and after the change.

C. Describe briefly the process by which the Domain Name System operates.

D. Briefly describe one way the Domain Name System was not designed to be completely secure.

E. Briefly describe one security concern and one coping strategy for the Domain Name System insecurity.

(http://csmatters.org) 3 - 6

0b11 - 0b110

Search Engines: Finding Information

Unit 3. Information and the Internet

Revision Date: Aug 18, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

This lesson investigates how search engines work: the spiders that crawl the web in search of valuable information, the data farms that store the data, and the processes used to organize current and historical data. The search process starts before you ever type a query, by crawling and indexing trillions of documents. Students will create a concept map illustrating their understanding of the operations of a search engine. A concept map is an artifact that could be created as part of the Explore Performance Task at the end of Unit 3.

Outcomes

Students will be able to:

- Describe the processes used by modern search engines to index content on the Internet
- Arrange the order of operations used in creating an index.
- Define basic search engine terms: spider, bot, crawl, data farm,
- Compare category based searching with indexed searching.
- Use an online tool to create a knowledge diagram of related information.

Overview

1. Getting Started (5 min) - Think-Pair-Share on Internet Searches
2. Activities (40 min) - Students cultivate an understanding of searching and build concept maps.
3. Wrap-Up (5 min) - Share ideas

Source

The slides for the guided exploration of search methods were adapted from slides provided by Marie desJardins at the University of Maryland, Baltimore County.

Learning Objectives

CSP Objectives

- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1H - Different algorithms can be developed to solve the same problem.
  - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.

7.1.1 G

7.3.1 AGJ

Math Common Core Practice:

- MP4: Model with mathematics.

Key Concepts

Students will understand the many processes that are required for an effective search engine.

Students will create diagrams and concept maps, do some investigations and discuss how search engines work, and then will individually use a computational tool to create an online diagram illustrating their understanding.

Essential Questions

- How can computing extend traditional forms of human expression and experience?
- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How can computation be employed to help people process data and information to gain insight and knowledge?
- How can computation be employed to facilitate exploration and discovery when working with data?
- What considerations and trade-offs arise in the computational manipulation of data?

- What opportunities do large data sets provide for solving problems and creating knowledge?
- How are algorithms implemented and executed on computers and computational devices?
- How are programs developed to help people, organizations or society solve problems?
- What is the Internet, how is it built, and how does it function?
- What aspects of the Internet's design and development have helped it scale and flourish?
- How does computing enhance human communication, interaction, and cognition?
- What are some potential beneficial and harmful effects of computing?

Teacher Resources

Student computer usage for this lesson is: **required**

- Powerpoint modified from Marie desJardins's UMBC search engine powerpoint ("About Search Engines" in Lesson Resources folder)
- Google's interactive story on how search works: the crawling and indexing video http://www.google.com/intl/en_us/insidesearch/howsearchworks/crawling-indexing.html (http://www.google.com/intl/en_us/insidesearch/howsearchworks/crawling-indexing.html) *(first 2 minutes only)*
- Google data centers https://www.google.com/about/datacenters/inside/ (https://www.google.com/about/datacenters/inside/)
- Online poster of the steps that happen in Google before and while you search: http://static.googleusercontent.com/media/www.google.com/en/us/intl/en_us/insidesearch/howsearchworks/assets/searchInfographic.pdf (http://static.googleusercontent.com/media/www.google.com/en/us/intl/en_us/insidesearch/howsearchworks/assets/searchInfographic.pdf)
- [Optional] Handout to guide student explorations (students can use their journals or their own papers if desired): "Handout: You and the Search Engine Diagram" in Lesson Resources folder
- [Optional] Handout for the Search Engine Treasure Hunt n Lesson Resources folder
- Online knowledge web concept map builder (see below) or Inspiration software installed on student computers.
    1. https://bubbl.us/ (https://bubbl.us/) 3 mind maps are free after creating a free acccount
    2. Insert a drawing into a Google doc (or create a Google diagram) https://support.google.com/docs/answer/179740?hl=en (https://support.google.com/docs/answer/179740?hl=en)
    3. http://www.softschools.com/teacher_resources/concept_map_maker/ (http://www.softschools.com/teacher_resources/concept_map_maker/) is a very simple, free tool that does not require an acount but is fairly limited.

Lesson Plan

Getting Started (5 min)

Students should journal on the following question:

"How many searches do you think are done each day using the Google search engine?"

Pair and share, then show this amazing live counter of internet searches: http://www.internetlivestats.com/google-search-statistics/ (http://www.internetlivestats.com/google-search-statistics/)

Guided Activities (40 min)

Activity 1 (25 min) - Understanding Search

Use the slide presentation "About Search Engines" (in Lesson Resources folder) to direct students through this lesson.

1. Students create a diagram of their best understanding of what happens when you type a query into a search engine. (Either provide the "Handout: You and the Search Engine Diagram" handout from the Lesson Resources folder or have students write on their own paper.)
2. Demonstrate how a search engine works with the video *(first 2 minutes only)* http://www.google.com/intl/en_us/insidesearch/howsearchworks/crawling-indexing.html (http://www.google.com/intl/en_us/insidesearch/howsearchworks/crawling-indexing.html) and diagram. Students can put a star next to each step in the process they thought of, then add to their diagrams to make them more complete.
3. Direct students to go to GoogleFight.com. (http://GoogleFight.com) [for saving time just have the teacher demonstrate] Discuss what happens. Ask:
    - Are all searches completed in the same amount of time?
    - Why or why not?
    - How does Google get the numbers to show on the results?
    - Are the numbers really an indication of the popularity of one thing vs another?
4. Students work in pairs and use a variety of search engines to find answers to a treasure hunt. The student worksheets are in the resource folder. Lead the discussion with "Socratic questioning." Start with what a URL is (URL = "Uniform Resource Locator" - a unique address of a document or resource on the Internet. http://docs.oracle.com/javase/tutorial/networking/urls/definition.html (http://docs.oracle.com/javase/tutorial/networking/urls/definition.html)). After that, branch out to other questions:
    a. Why are there different search engines?

b. Why do people get different answers?
c. Is it better to type in a whole question or just to pick keywords?
d. What do you do when you don't get the answer you want the first time you search?

5. Emphasize how the Intenet is changing. Use the lesson presentation or other resources.
   1. http://www.hypebot.com/hypebot/2012/07/see-how-much-the-internet-has-changed-the-music-industry-infographic.html (http://www.hypebot.com/hypebot/2012/07/see-how-much-the-internet-has-changed-the-music-industry-infographic.html)
   2. http://www.digitaltrends.com/computing/cisco-internet-traffic-966-exabytes-per-year-in-2015/ (http://www.digitaltrends.com/computing/cisco-internet-traffic-966-exabytes-per-year-in-2015/)
6. Discuss the current state of the Internet, how complicated it is, and why it can't be indexed completely every day (think size). How is it possible to keep track of such a huge volume of data?

Activity 2 (15 min) - Concept Map Creation

Have students create a concept map of ideas relating to search engines, doing additional research to round out their understanding. (See Teacher Resources for online tools that can be used to create concept maps.)

Wrap Up (5 min)

Share ideas from the students' concept maps. Point out that the concept map (if done online) is an artifact that was created using a computer to present information visually.

Optional Extension: (for fast moving classes who need more to do)

Google tracks everything that everyone queries. (Is this an invasion of your privacy?) The results are fascinating.

Look at www.google.com/trends (http://www.google.com/trends). You can look at trends by region and limit them to a date and/or place. For example search for "Obama, McCain" limiting your search to 2008, and the United States. What conclusions do you draw?

Pick another topic of interest to explore in Google trends to reveal society's interests.

Options for Differentiated Instruction

Students can create diagrams and concept maps on paper by hand if that is helpful.

Be sure to assign roles to pairs when working together. Don't allow one partner to be passive while the other is active.

Evidence of Learning

Formative Assessment

Students create a concept map of what they learned with additional research on the topic.

Summative Assessment

Students will develop a visual diagram of the processes involved in indexing the Internet by a search engine.

(http://csmatters.org) 3 - 7

0b11 - 0b111

Search Engines: Page Rank and Retrieval

Unit 3. Information and the Internet

Revision Date: Jul 18, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary

This lesson has two main objectives.

The first focuses on search engine algorithms and the impact search engines have on our lives. Search engine page rank algorithms rely on many factors to predict what someone is looking for. The business advantage of appearing on the front page of a Google search is tremendous. However, as more information is tracked about our interests and preferences in order to customize the results of our searches, we have to ask whether or not the loss of privacy is worth the results.

The second objective is to introduce students to creating a visual artifact (knowledge required for performance tasks). Students will research a page ranking subtopic, prepare a one minute speech, and (if possible) create a video to accompany the speech.

Outcomes

- A presentation guides the discussion of how search engines work, what page rank is, and how results differ for a variety of reasons.
- Students then do a quick research project to gather information on assigned, related topics
- Students create a 1-minute speech and presentation on their chosen research topic. If the classroom has the equipment, they will create a video artifact on their topic to share with the class either in the classroom or as homework posted online

Overview

Session One

1. Getting Started (5 min) - Journaling on online search methodology
2. Activity (45 min) - Students discuss page rank and begin research for presentations

Session Two

1. Getting Started (5 min) - Journaling on search engine data mining and introduce presentation activity
2. Activity (35 min) - Create Presentations/Videos on previously researched topics.
3. Presentation of videos or talks (10 min)
4. Optional Homework: Have students watch the rest of the videos made and write about what they learned.

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
- 1.2.1 - Create a computational artifact for creative expression.
  - 1.2.1A - A computational artifact is something created by a human using a computer and can be, but is not limited to, a program, an image, an audio, a video, a presentation, or a Web page file.
  - 1.2.1B - Creating computational artifacts requires understanding of and use of software tools and services.
  - 1.2.1E - Creative expressions in a computational artifact can reflect personal expressions of ideas or interests.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
- 1.3.1 - Use computing tools and techniques for creative expression.
  - 1.3.1E - Computing enables creative exploration of both real and virtual phenomena.
- 3.2.1 - Extract information from data to discover and explain connections or trends
  - 3.2.1D - Search tools are essential for efficiently finding information.

3.2.1. D

Key Concepts

Students will understand that the page rank algorithm depends on many factors, has changed over time, and has a large impact on the traffic that a site gets.

Students will give examples of how their activity online is tracked and how the knowledge of them is used to taylor the results and the possible repercussions.

Students will create an artifact using screen capture of themselves discussing and analyzing an aspect of searching.

Essential Questions

- How can computing extend traditional forms of human expression and experience?
- How can computation be employed to help people process data and information to gain insight and knowledge?
- What is the Internet, how is it built, and how does it function?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- "Search Engine Background" NOTE: This document explains the content of each slide in the presentation WITH answers to the questions in the presentation.
- "PageRank" - slide presentation
- Handouts for students (can be placed on the student's drives or printed out on paper):
  - "PageRank Student Handout" (optional notes to go along with the PowerPoint - gives students a place to answer questions posed in the presentation)
  - "1 minute talk directions.odt" (to help students organize their video)
  - "Sample 1 minute script on keyword matching"
- "Sample 1 minute video artifact on keywords.swf" (1:15)

Online Videos:

- "How Search Works by Matt Cutts" https://www.youtube.com/watch?v=BNHR6IQJGZs (https://www.youtube.com/watch?v=BNHR6IQJGZs) (review of spiders, how they follow links and fetch the pages to index. followed by page rank and spam avoidance) (3:14)

Sites Used in this Lesson:

- Screencast-O-Matic http://screencast-o-matic.com/ (http://screencast-o-matic.com/) Used by students to create an video artifact of their screen to accompany their speech.
- Jing http://www.techsmith.com/jing.html (http://www.techsmith.com/jing.html) Free download for screen capture to create artifact to accompany speech.
- PageRank Checker http://checkpagerank.net/ (http://checkpagerank.net/) Students use this page to check the page rank and other information about web sites.

Lesson Plan

Getting Started (5 min)

Students should take a few minutes to journal on the following question:

Which are you more likely to do if you don't see an answer to a search request on the first page: click forward to page 2 of the results or ask the question differently? Why?

(Encourage students to discover that it is very valuable to a business to appear at the top of the search engine rankings and that often thousands or millions of results are returned in a single search.)

Activity (45 min)

Part 1 (25 min) - How Search Engines Work

(Use the PageRank presentation in the lesson folder to guide discussion.)

Note: Guidelines for the teacher are in the "Teacher Notes on PageRank Presentation" document. This document also contains an answer key. (*Students can record their notes in the "PageRank Student Handout".*)

1. Watch the three-minute video on Google search closely to pick up details. Pause, take notes, and discuss as needed.
2. Allow students to generate ideas on why one webpage might have a higher PageRank than the other. [slide 3]
3. Look at the HTML code of the webpage in the PowerPoint to discover the frequency of keywords including synonyms, and occurrence in titles and metatags. [slide 4] (*Student handout also has a printout of the HTML code for students to get a closer look.*)
4. Assign words to students to define. Share definitions with the class. [slide 5]
   - Backlink
   - Referring domain
   - PR Quality

- SEO
- Alexa Rank
- Directory listed
- Domain age.
- Whois

5. Discuss possible reasons why two different people can get different results doing the same search.

Part 2 (20 min) Preliminary Research for creation of video artifacts/PowerPoints

1. Assign topics for research to student groups. (There are additional topics in the "Search Engine Background" information document if desired). Here are several suggestions:
   - What are additional factors in page rank?
   - What do people do to achieve SEO? (search engine optimization)
   - What is Google bombing? How does it work?
   - How much storage is needed to store Google's index? How many server farms are needed to store it all? What is the design philosophy of server farms?
   - What's the environmental impact of server farms? How do they try to stay green?
   - How does advertising affect search engines? Is it necessary? What is "pay per click" and "click fraud"?
   - How is Google getting good at finding things like pictures, videos and other kinds of information beyond just words?
   - How do directories work? Show some examples, such as https://dir.yahoo.com/ (https://dir.yahoo.com/) .
2. **If you have video recording equipment:** Demonstrate how to create a high-quality video artifact (the kind students might choose to create for their Performance Task).
   - Student handout: 1 minute talk directions -- go over this handout with them.
   - Show the Sample 1 minute video artifact on keyword matching (in the Lesson Resources folder).
3. With the remaining time, have students begin their research on their chosen topic.

Session 2

Getting Started (5 minutes)

Journal (3 min)

Why could it be beneficial for a search engine to keep track of what people are searching for? Discuss.

(Possible answer to lead students toward: Topics sporadically become popular, and knowing what results people like can make it easy to suggest sites to others looking for similar things. History data can also enable a search engine to suggest a search phrase when a single word or only a few letters are typed in.)

Introduction **(2 min)**

Explain that students will be creating a presentation on the topic they researched in the last session. This presentation should be scripted, and make use of a PowerPoint and sources from the internet. They will have 30 minutes to make this presentation. (Slide 8 is made for video creation, but works well for general presentations too.)

For classes with enough video recording equipment for all groups:

Explain that students will create their own video explanations of how one feature of search engines works. Go over the "1 minute talk directions.odt" together to help students organize their video.

Activity (35 minutes)

Students should split into their groups and begin work. Allow only 10 minutes for additional research as needed. They will take the remaining 25 minutes to:

a. Make a PowerPoint, gather search results to use as examples, create a rough script, and practice their presentation.
b. **If they are making a video:** Write a script, either create a PowerPoint to voiceover **or** choose some search results to analyze, and practice. The last 5-10 minutes should be used to record a 1 minute video clip of their presentation allowing for multiple retakes)

Presentation of videos or talks (10 minutes)

Show as many videos/ group presentations as you can share with the class. If there are videos, assign the remainder to be watched as homework and have students bring in notes on the key points learned from each video.

Options for Differentiated Instruction

For a shorter class, don't have students take notes, just discuss the slides.

For the Explore performance task, each student should be able to create their own artifact. You could have the students work on the presentations individually in this lesson, as a practice for the Explore task, if your class is fairly competent with the technology. For students with less experience (or to save time during presentations), it could be beneficial to have students create these artifacts in pairs, with some pairs repeating topics for comparison.

Evidence of Learning

Formative Assessment

Students share best definitions of page rank related terms

Students analyze web pages for reasons for differences in page rank

Summative Assessment

Students create a one-minute video clip on a topic related to the operation of search engines.

Internal Use Only

Pending Tasks

Diane O'G said she would update this lesson to include the video generation process (day 2)

(http://csmatters.org) 3 - 8

0b11 - 0b1000

Optional

Basic Statistics with Excel

Unit 3. Information and the Internet

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary: This lesson is designed for students to review basic statistics, including calculations of the mean, median, mode, and standard deviation. It will also give the students some experience using spreadsheet software to calculate the statistics and to create histograms. *Note: This lesson is intended primarily as a review and a reminder of material that should already be familiar to the students. If your students have little familiarity or experience with using Excel to compute statistics or generate plots, you may wish to extend this lesson to two sessions, and provide more scaffolding and instruction on the basic mechanisms.*

Outcomes:

- Students will review the basic statistical concepts of mean, median, mode, and standard deviation.
- Students will use spreadsheet software to calculate the statistics and to create histograms.

Overview:

1. Getting Started (5 min)
2. Introduction of Content (10 min) - Statistics Introduction and Review
3. Guided Activity (30 min) - Students Create Plots and Calculate Candy Statistics
4. Wrap Up (5 min) - Journal

Source: This lesson was adapted from Unit 2: The Engineering Design Process, Lesson 2: Collecting and Processing Information ©2013 International Technology and Engineering Educators Association Foundations of Technology, Third Edition/ Technology, Engineering, and Design

Learning Objectives

CSP Objectives

- 3.1.1 - Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.
  - 3.1.1D - Insight and knowledge can be obtained from translating and transforming digitally represented information.
  - 3.1.1E - Patterns can emerge when data is transformed using computational tools.
- 3.1.3 - Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.
  - 3.1.3A - Visualization tools and software can communicate information about data.
  - 3.1.3B - Tables, diagrams, and textual displays can be used in communicating insight and knowledge gained from data.
  - 3.1.3C - Summaries of data analyzed computationally can be effective in communicating insight and knowledge gained from digitally represented information.
  - 3.1.3D - Transforming information can be effective in communicating knowledge gained from data.
- 3.2.1 - Extract information from data to discover and explain connections or trends
  - 3.2.1A - Large data sets provide opportunities and challenges for extracting information and knowledge.
  - 3.2.1B - Large data sets provide opportunities for identifying trends, making connections in data, and solving problems.
  - 3.2.1C - Computing tools facilitate the discovery of connections in information within large data sets.

3.1.1 DE

3.1.3 ABCD

Vocabulary

Statistics: the collection and organization of data (science) as well as the analysis and presentation of those data (mathematics).

Mean: the average of a given data set.

Median: the middle number in a given data set.

Mode: the most frequently occurring number in a given data set.

Standard Deviation: how much variation exists from the average (mean) in a given data set.

Range: the difference between the largest and smallest values in a given data set.

Math Common Core Practice:

- MP5: Use appropriate tools strategically.

Common Core Math:

- S-ID.1-4: Summarize, represent, and interpret data on a single count or measurement variable

Common Core ELA:

- RST 12.10 - Read and comprehend science/technical texts

NGSS Practices:

- 5. Using mathematics and computational thinking

Key Concepts

The students must understand the basic statistical concepts of mean, median, mode, and standard deviation. They must also be able to use spreadsheet software to calculate the statistics and to create histograms.

Students often have some initial difficulty learning how to use formulas in the spreadsheet software to do the calculations.

Essential Questions

- How can computation be employed to help people process data and information to gain insight and knowledge?
- How can computation be employed to facilitate exploration and discovery when working with data?

Teacher Resources

Student computer usage for this lesson is: **required**

For Each Student:

- Package of colored candy. Alternatively, you can ask students at the end of the previous lesson to collect some distributional data to use for the exercise (ideas: colors of cars in the school parking lot, colors of shirts worn by students in the room, favorite sports teams or bands of the students), or you can simply provide some data for the students to use. This could be either representing colors of candy (using the test data in the lesson if you would like), or similar distributional statistics.
- Excel software or other spreadsheet software
- Web resource with information about measures of central tendency: https://statistics.laerd.com/statistical-guides/measures-central-tendency-mean-mode-median.php (https://statistics.laerd.com/statistical-guides/measures-central-tendency-mean-mode-median.php)

Lesson Plan

Getting Started (5 min)

- Students should describe what they know about statistics in their journals.
- Have students share what they know about statistics and introduce the lesson.

Introduction of Content (10 min)

Review of Statistics:

Present a review of basic statistics (min, max, mean, median, mode, and range), and use the following board exercise to have the class review their understanding of these basic concepts:

- Ask eight or so randomly selected students for their birth date (day of the month).
- Write these numbers on the board.
- On the side of the board, list the key terms "min," "max," "mean," "median," "mode," and "range."
- Ask the class as a group to compute each of these values:
  - **Min:** The smallest number (what if there is more than one? - no problem!)
  - **Max:** The largest number (what if there is more than one? - no problem!)
  - **Mean:** The average (sum of the numbers, divided by how many numbers there are)
  - **Median**: The center value in a sorted list of numbers.
    1. Have the students help you to rewrite the values from smallest to largest.
    2. Which is the middle number?
    3. Since there are 8 numbers, there is no middle number!
    4. In this case, the median is the mean (average) of the two center numbers = 4th number + 5th number / 2.
  - **Range:** The difference between the largest and smallest value (max - min).
  - **Mode:** The most frequently appearing value. In such a small set, there is likely to not be a mode, unless two students happen to share the same birth date. You might wish to poll the students for another number (e.g., the students' grade) that's likely to have more repeated values, and then compute the mode (and, optionally, the other statistics).

Discussion:

Ask the class to come up with situations where it might be most useful to compute the mean, median, or mode of a set of values. Encourage them to understand that each of these statistics can be useful in different situations, but may be misleading. Have them generate sets of data that would give "misleading values" for mean (if there is an "outlier value"), median (if the values have a longer "tail" on one side than the other), or mode (if there is a frequent value that happens to occur at one end or the other of a wider range).

- This web resource has a good discussion of different measures of central tendency and when they are appropriate: https://statistics.laerd.com/statistical-guides/measures-central-tendency-mean-mode-median.php (https://statistics.laerd.com/statistical-guides/measures-central-tendency-mean-mode-median.php)

Guided Activity (30 min) - Candy Statistics

Note: The teacher may want to do this activity along with the students, displaying the spreadsheet on a screen so that the students may ask questions and see how to do the statistical calculations using the spreadsheet software. Students who do not have much experience with spreadsheets may need more scaffolding and instruction. (If you have many such students, you may wish to spread this lesson out over two class sessions.)

Students will use spreadsheet software, such as Excel, to calculate the average number and standard deviation of candy color in an individual-sized bag of M&Ms, Skittles, or other colored candy. Optionally, students may compare their results to other online published statistics for each candy.

1. Have the students predict how many individual candy pieces are in their bag of candy and write their predictions in their journals.
2. Have the students open their bag of candy and sort the candy into categories based on color.
3. Have the students note the difference in the total number of candies predicted versus the actual number that was in the packet. They should note the difference in their journals.
4. Open an Excel program and create a spreadsheet like the following. Each Trial Number in the example below corresponds to a student or group in the class. *(**Note:** If you do not have candy to do the counting exercise, you may simply give the sample spreadsheet below to the students.)*

Candy Statistics						
Trial Number	1	2	3	4	5	6
Yellow	17	20	24	19	19	17
Red	21	13	19	21	15	18
Blue	10	18	16	18	21	20
Brown	7	12	5	12	12	14
Green	26	26	16	17	22	18
Orange	24	16	20	15	15	16
Package Total	105	105	100	102	104	103

The students will also need to create columns further to the right labeled Mean, Median, Mode, and Standard Deviation.

Mean	Median	Mode	Standard Deviation
19.375	19	19	2.199837656
18.125	19	19	2.799872446
17	17.5	18	3.338091842
9.5	10	12	3.380617019
21.125	22	22	3.833592124
18.25	17.5	16	3.284161124

5. Each student will enter their own data for each color and the data from another student or group into the table.
6. Using their data, students will:
    1. Calculate the mean value for each color category within the experiment. They should use the Average function to do the calculation.
    2. Calculate the median, mode, and standard deviation for all color categories. They should use the appropriate functions to do the calculations.
    3. Calculate the package total for each trial by using the `SUM` function.
    4. Create a ± 3σ histogram for each candy color.
    5. Create a frequency distribution table for each candy color, as illustrated below.
    6. Create a histogram for each candy color, using your bin and frequency data.

Yellow Candy σ =	2.199837656Get on
3Cs	25.97451297
2S	23.77467531
1s	21.57483766
Mean	19.375
-1s	17.17516234
-2s	14.97532469
-3s	12.77548703

Wrap Up (5 min)

Students will answer the following question in their journals:

- Why is it important to use statistics to understand large data sets? When are different measures of central tendency appropriate or inappropriate?

Options for Differentiated Instruction

Learners may be paired to assist each other in the use of the spreadsheet software.

Evidence of Learning

Formative Assessment

The teacher should frequently check the students' work for accuracy as the lesson progresses so that misunderstandings may be quickly resolved.

Summative Assessment

- Have the students calculate the mean, median, mode and standard deviation of a set of data.
- Have the students use a spreadsheet to do statistical calculations and create a histogram.

(http://csmatters.org) 3 - 9

0b11 - 0b1001

Practice for Explore Performance Task

Unit 3. Information and the Internet

Revision Date: Jul 22, 2016 (Version 2.0)
Duration: 3 50-minute sessions

Lesson Summary

Summary

This lesson provides the students with an opportunity to practice the AP CSP Explore Performance Task with a given set of tasks from which they may choose. Students will use a rubric to perform self-assessment of work generated for the Explore Performance Task.

Outcomes

- Students will practice the AP CSP Explore Performance Task.
- Students will use a rubric to grade the paper created in the previous class to evaluate their own work and make modifications where necessary.

Overview

Session 1

1. Getting Started (5 min)
2. Introduction of Content (10 min)
3. Independent Activity (30 min)
4. Wrap Up (5 min)

Session 2

1. Getting Started (5 min)
2. Independent Activity (40 min)
3. Wrap Up (5 min)

Session 3

1. Getting Started (5 min)
2. Independent Activity (40 min)
3. Wrap Up (5 min)

Source: The lesson models the College Board AP CSP Explore Performance Task.

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
  - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.1 - Create a computational artifact for creative expression.
  - 1.2.1A - A computational artifact is something created by a human using a computer and can be, but is not limited to, a program, an image, an audio, a video, a presentation, or a Web page file.
  - 1.2.1C - Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming integrated development environments (IDEs), spreadsheets, three-dimensional (3-D) printers, or text editors.
  - 1.2.1E - Creative expressions in a computational artifact can reflect personal expressions of ideas or interests.
- 1.2.5 - Analyze the correctness, usability, functionality, and suitability of computational artifacts.
  - 1.2.5A - The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact.
- 3.3.1 - Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
- 7.4.1 - Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts.

7.3.1 A

7.4.1 ABE

Common Core ELA:

- RST 12.1 - Cite specific textual evidence to support analysis of science and technical texts, attending to important distinctions the author makes and to any gaps or inconsistencies in the account.
- RST 12.6 - Analyze the author's purpose in providing an explanation, describing a procedure
- WHST 12.1 - Write arguments on discipline specific content
- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.9 - Draw evidence from informational texts to support analysis, reflection, and research

NGSS Practices:

- 7. Engaging in argument from evidence

NGSS Content:

- HS-ETS1-3. Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs that account for a range of constraints, including cost, safety, reliability, and aesthetics as well as possible social, cultural, and environmental impacts.

Key Concepts

The students will be able to use reliable research findings to generate a computational artifact and respond to questions on that innovation.

Essential Questions

- How can a creative development process affect the creation of computational artifacts?
- How can computing and the use of computational tools foster creative expression?
- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How can computation be employed to help people process data and information to gain insight and knowledge?
- What considerations and trade-offs arise in the computational manipulation of data?
- What aspects of the Internet’s design and development have helped it scale and flourish?
- How does computing enhance human communication, interaction, and cognition?
- How does computing enable innovation?
- What are some potential beneficial and harmful effects of computing?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

Teacher Resources

Student computer usage for this lesson is: **required**

In the Lesson Resources folder:

- "Explore_Performance_Task_Rubric_Nov2015" : AP test rubric for the overall practice performance
- "Practice Explore Task Job Completion Check List" : check list for task jobs
- "Practice Explore Task Response Document November2015" : response document
- "CSMattersWritingRubric.docx" : detailed formative assessment writing-specific rubric for the practice Explore performance task

Lesson Plan

Session 1:

Getting Started (5 min)

- Previous evening's homework was to complete the Practice Explore Performance Task worksheet.
- Group students by topic in groups of 3 – 4. Have students share findings with each other and report any missing items. Teacher will need to circulate to assist where needed.

Introduction of Content (10 min)

Instruct the students that they are to use their findings to generate a computational artifact and a one-page paper on an innovation.

Directions for artifact:

The artifact is an original digital artifact screencast or knowledge map diagram that you create to express the effects of your chosen innovation.

Directions for the Paper:

Students are to generate a written document in which they respond directly to the following prompts. Their document should be a one-page paper that may include illustrations.

- Describe the area of our lives (social, economic, or cultural) that has been most impacted by the innovation, and discuss the significance of the innovation to this area, using references to support your argument.
- Describe the population that is affected by the innovation and explain why that population is significant.
- Describe the connection between your artifact and the innovation you explored.
- Identify and describe how information sharing has affected this innovation.
- Discuss the extent to which this innovation is dependent on the Internet.
- Describe any security concerns and explain how they relate to the innovation.
- Describe the beneficial as well as any harmful effects of the innovation you explored.

Independent Activity (30 min)

Writing responses to questions posed on "Practice Explore Task Response Document" and creating artifact.

Remind students that brevity is important for the performance tasks; it is a talent to be able to get a message across with real content succinctly.

I have made this letter longer than usual, only because I have not had time to make it shorter. ~Blaise Pascal (1623-1662).

Wrap Up (5 min)

Have students complete the "Practice Explore Task Job Completion" form, indicating where they are in the process of the Practice Artifact.

Homework

Complete any additional research needed to complete the document (if needed).

Session 2:

Getting Started (5 min)

The previous evening's homework was to complete the paper that students were working on in the previous class. Pass out the rubric and have students go through their paper to verify that they have all points covered. Point out that most of the points awarded are based on the report (75% of the total score) and the artifact represents 25% of the score. Teacher will need to circulate to assist where needed.

Independent Activity (40 min)

- Students are to make any modifications necessary on their paper and work on the artifact. The entire project will be collected next class.
- The artifact is an original digital artifact screencast or a knowledge map diagram that you create to express the effects of your chosen innovation.

Wrap Up (5 min)

Have students complete a Job Progress form indicating where they are in the process of the Practice Artifact.

Homework

Complete the paper using the rubric as a guide. Complete the plan for the artifact; you will have one more class period to work on the artifact before the project is due.

Session 3:

Getting Started (5 min)

The previous evening's homework was to complete responses to "Practice Explore Task Response Document" modifications as indicated using the rubric that students received in the previous class. Using the artifact part of the rubric, have students go through their plans for the artifact to verify that they have all points covered. Point out that most of the points awarded are based on the report (75% of the total score) and the artifact represents 25% of the score. Teacher will need to circulate to assist where needed.

Independent Activity (40 min)

The responses to "Practice Explore Task Response Document" for the Practice Explore Performance Task should be complete. Students should be focused on the artifact. The entire project will be collected next class.

Wrap Up (5 min)

Have students complete a Job Progress form indicating where they are in the process of the Practice Artifact.

Homework

Complete artifact using the rubric as a guide. The Practice Explore Performance Task is due at the start of the next period.

Options for Differentiated Instruction

SPED/LLD: need to be frequently monitored and assisted as needed.

Teachers may choose to spread this activity out into multiple class sessions over a longer period of time, to give students some more time to work on the paper and artifacts at home (especially if the class meets every day, since "overnight homework" is often difficult for students to complete effectively when they are involved in other after-school activities and have assignments for other classes).

Evidence of Learning

Formative Assessment

Practice Explore Performance Task -- provide feedback, using College Board's rubric (Explore_Performance_Task_Rubric_Nov2015.pdf and the CSM detailed Explore Task writing rubric (CSMattersWritingRubric.docx and CSMattersWritingRubric.pdf) -- both can be found in the lesson folder

Summative Assessment

Explore Performance Task

(http://csmatters.org) 3 - 10

0b11 - 0b1010

Cybersecurity: Attacks, Protection, and Impact

Unit 3. Information and the Internet

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Pre-lesson Preparation

This lesson does not require computers, but teaching this lesson without computers would require printing the necessary articles and providing textbooks or printed articles about the various cyber-attacks.

Summary

Reflecting on the fact that the Internet was not designed with security in mind, students will examine the devastating impact of cyber attacks. Students will study types of cyber attacks and the vulnerabilities they exploit, and identify the roles of software, hardware, people, and the Internet. Students will identify potential cybersecurity concerns in systems built on the Internet.

Outcomes

- Students will understand types of security violations.
- Students will understand types of protections.
- Students will compare negative impacts of different types of attacks.

Overview

1. Getting Started (5 min) - Discussing the internet's security concerns.
2. Guided Activities (40 min) - Students explore and research specific cybersecurity attacks and their impacts.
3. Wrap Up (5 min) - Journaling on the accessibility of student data.
4. Homework - Research anti-virus software.

Learning Objectives

CSP Objectives

- 6.3.1 - Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it.
  - 6.3.1A - The trust model of the Internet involves trade-offs.
  - 6.3.1B - The DNS was not designed to be completely secure.
  - 6.3.1D - Cyberwarfare and cybercrime have widespread and potentially devastating effects.
  - 6.3.1E - Distributed denial-of-service attacks (DDoS) compromise a target by flooding it with requests from multiple systems.
  - 6.3.1F - Phishing, viruses, and other attacks have human and software components.
  - 6.3.1G - Antivirus software and firewalls can help prevent unauthorized access to private data.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
  - 7.1.1M - The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.
  - 7.1.1O - The Internet and the Web have impacted productivity, positively and negatively, in many areas.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
  - 7.3.1A - Innovations enabled by computing raise legal and ethical concerns.
  - 7.3.1G - Privacy and security concerns arise in the development and use of computational systems and artifacts.
  - 7.3.1L - Commercial and governmental curation of information may be exploited if privacy and other protections are ignored.

Common Core Math:

- S-IC.1-2: Understand and evaluate random processes underlying statistical experiments

Common Core ELA:

- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.9 - Synthesize information from a range of sources
- RST 12.10 - Read and comprehend science/technical texts
- WHST 12.1 - Write arguments on discipline specific content

Key Concepts

6.3 Cybersecurity is an important concern for the Internet and the systems built on it.

The Internet was not built with security in mind, leaving computers vulnerable to cyber attacks. This makes cybersecurity an extremely important concern when designing and implementing systems that are built on the Internet. Students need to be able to identify potential problems that could arise and potential options for protecting against these problems.

Essential Questions

- How is cybersecurity impacting the ever increasing number of Internet users?
- How does computing enable innovation?
- What are some potential beneficial and harmful effects of computing?

- How do economic, social, and cultural contexts influence innovation and the use of computing?

Teacher Resources

Student computer usage for this lesson is: **optional**

In the Lesson Resources folder:

- "Cyber Security" : slides for instruction during the whole class
- "Cyber Attacks News Articles" : the list of news articles about real life cyber attacks for teachers (with instructions)
  - The diagram for the sticky note activity is in this document
- "Cyber Attacks Notes WS" : worksheet for students to use in taking notes on different types of attacke

Journal Sample Response:

- "The Internet was originally designed to be used by a group of people who trusted each other. This means that it was not built with security in mind, but rather openness and sharing. Now that anybody can access the Internet, users cannot trust everybody else they are connected to. This means that security measures must be put in place to protect users and systems."

Example for Presentations:

Information to present about firewalls. (Included in the slides)

"You can protect against certain attacks. One way to protect against them is a firewall."

1. Where did the name come from?
    1. We have physical firewalls in school (and other buildings) that are designed to open to let people in and out, but close to keep fire contained (don't let it through)
2. How does it work? Describe the process, making sure to note the role of each of the following: (not all will necessarily apply)
    1. A *firewall* is installed to be a barrier between a computer (or local network) and the **Internet**. A **person** has to purchase / install the firewall to protect their system. Firewalls can be **software** or **hardware** and sometimes people use both. Firewalls examine the packets attempting to go in or out from the computer (or local network) to/from the Internet. It can keep attacks like viruses out, and keep sensitive or private data in.
3. Visual from https://mdilog.com/help/security (https://mdilog.com/help/security)

Lesson Plan

(**Note**: There is a PowerPoint to be used with this entire lesson: **"Cyber Security Lesson Slides"** in the Lesson Resources folder.)

Getting Started (5 min)

In their journals or as a class, students should discuss the following:

1. Describe the "trust model" that the Internet was originally designed upon.
    - The "trust model" was introduced in Lessons 3-4, 3-5, and 3-6 that introduced the Internet.
2. List the problems with using the trust model, now that anybody can access the Internet.

Guided Activities (40 min)

Part 1 (20 min) - Readings

1. Each student will read a short article from the news about a specific attack that took place and identify the type of attack. Through their readings, the students will identify the negative effects of the attack. (A list of possible articles is in the "Cyber Attacks News Articles" document in the lesson folder.)
2. Students will use sticky notes to record the following information (at least one per student):
    1. The student's name
    2. The type of cyber attack
    3. The impact of the cyber attack
3. On the board, the teacher will set up a space with the types of attacks on the y-axis, and the level of impact on the x-axis (see teacher resources). Students will place the sticky notes on the diagram where they think it fits.
4. Students will be asked what information and knowledge they can draw from the diagram.
5. The teacher will model the next activity for the students by presenting some information on firewalls. This includes explaining where the name "firewall" comes from, how a firewall works, the roles of software, hardware, people, and the Internet, and a visual representation.

Part 2 (20 min) - Small Group Activity

1. Students will be grouped by the type of attack they read about. They will conduct research to answer the following questions: (some resources will be provided, but students can also search for others. If no computers are provided, it will be up to the teacher to find these

additional resources)

1. Where did the name come from?
2. How does the attack work? Describe the process, making sure to note the role of each of the following: (not all will necessarily apply)
    1. The Internet
    2. Software
    3. Hardware
    4. People
3. Find or create a visual that illustrates the attack OR act out the process.

2. Each group (or at least some, depending on time) will present their findings to the class in 2 minutes or less.
    1. Students should use the "CyberSecurity Notes WS" document to take notes for use in studying for the Unit 3 assessment.

Wrap Up (5 min)

Students will consider the following prompt, and record their thoughts in their journals:

What possible problems are there with the fact that student data (including your courses, grades, attendance, home address, and birthdate) is stored in a database that is easily accessible to teacher, administrators, and other staff from any computer connected to the Internet?

- What security concerns does this raise?
- What can be done to protect student data?

Homework:

Real World Connection: Protecting your Computer

Choose one of the following articles to read, based on the operating system you have running on one of your home computers, or the computer you normally use.

- Windows: http://windows.about.com/od/maintainandfix/a/4-Questions-To-Determine-If-Your-Windows-Pc-Is-Secure.htm (http://windows.about.com/od/maintainandfix/a/4-Questions-To-Determine-If-Your-Windows-Pc-Is-Secure.htm)
- Mac: http://www.pcmag.com/article2/0,2817,2408623,00.asp (http://www.pcmag.com/article2/0,2817,2408623,00.asp)
- Linux: http://www.zdnet.com/blog/btl/five-tips-for-improving-linux-security/35798 (http://www.zdnet.com/blog/btl/five-tips-for-improving-linux-security/35798)

Answer the following questions:

1. Does the computer have anti-virus software installed?

If yes, answer the following questions:

1. What is the name of the anti-virus software installed on the computer?
2. Is the anti-virus software on the computer up to date?
3. What features does the anti-virus software provide?

If no, do the following:

1. Find at least two different anti-virus programs for your operating system (one that is free and one that you must purchase).
2. Compare and contrast the anti-virus programs based on the features that they offer.
3. Talk to your parent about installing anti-virus software on your computer if you own one.
4. Does the computer have a firewall enabled?
5. Is the operating system up to date? Which version of the operating system is the computer currently running?
6. What other security measures have been taken to protect the computer?

Optional: Use this extended checklist to enhance the security of your computer.

http://m.wikihow.com/Secure-Your-PC (http://m.wikihow.com/Secure-Your-PC)

Evidence of Learning

Formative Assessment

The teacher will see where the students place the cyber attacks as they read about them on the impact graph and give appropriate feedback.

The teacher will monitor the research on cyber attacks and check for accurate information.

The teacher will clarify misconceptions that become evident during the group presentations.

Summative Assessment

Students will complete a journal entry by responding to questions about their personal and school related data being accessible through the Internet.

Internal Use Only

Pending Tasks

I am not sure how to modify Nik's request to be notified when links are no longer good in his large list of articles (where one is assigned to each student). I left the note about it in teacher resources, and I left the files needed in the Lesson Resources folder, but as it is, it will not notify anyone. -- Jan

(http://csmatters.org) 3 - 11

0b11 - 0b1011

Cryptography: Symmetric Encryption

Unit 3. Information and the Internet

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

Students are introduced to the topic of cryptography and learn to perform two encryption techniques. The students will identify the role of the algorithm and key in the encryption process. Students will use abstraction to see the general process used in symmetric encryption. The students will consider the strength of ciphers and the importance of keeping the key a secret.

Outcomes

- Students will understand how encryption is used to keep data secure.
- Students will learn how encrypting and decrypting data is accomplished using an algorithm and a key.
- Students will understand why the key must be kept a secret.

Overview

1. Getting Started (5 min) - Journal
2. Introduction to Content (15 min)
    1. Lesson Motivation [5 min]
    2. Presenting the Key Concepts [10 min]
3. Guided Activities (25 min)
    1. Practice [15 min]
    2. Follow Up: Analyzing the Strength of Ciphers [5 min]
    3. Follow Up: Defining Symmetric Encryption and Seeing the Abstraction in Symmetric Encryption Systems [5 min]
4. Wrap Up (5 min) - Journal

Learning Objectives

CSP Objectives

- 4.1.1 - Develop an algorithm for implementation in a program.
    - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
    - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.

    - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
    - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
    - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
    - 4.1.1H - Different algorithms can be developed to solve the same problem.
    - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
    - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
    - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
    - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
    - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
    - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
    - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
    - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.
    - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.2.1 - Explain how programs implement algorithms.
    - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
    - 5.2.1B - Program instructions are executed sequentially.
    - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
    - 5.2.1E - Program execution automates processes.
    - 5.2.1J - Simple algorithms can solve a large set of problems when automated.
- 5.3.1 - Use abstraction to manage complexity in programs.
    - 5.3.1A - Procedures are reusable programming abstractions.
    - 5.3.1B - A procedure is a named grouping of programming instructions.
    - 5.3.1C - Procedures reduce the complexity of writing and maintaining programs.
    - 5.3.1D - Procedures have names and may have parameters and return values.
    - 5.3.1E - Parameterization can generalize a specific solution.
    - 5.3.1F - Parameters generalize a solution by allowing a procedure to be used instead of duplicated code.
    - 5.3.1G - Parameters provide different values as input to procedures when they are called in a program.
    - 5.3.1H - Data abstraction provides a means of separating behavior from implementation.
    - 5.3.1I - Strings and string operations, including concatenation and some form of substring, are common in many programs.
    - 5.3.1J - Integers and floating-point numbers are used in programs without requiring understanding of how they are implemented.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
    - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.
    - 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
    - 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
    - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
    - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
    - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
    - 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.
- 6.3.1 - Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it.
    - 6.3.1A - The trust model of the Internet involves trade-offs.
    - 6.3.1H - Cryptography is essential to many models of cybersecurity.
    - 6.3.1I - Cryptography has a mathematical foundation.
    - 6.3.1K - Symmetric encryption is a method of encryption involving one key for encryption and decryption.

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP8: Look for and express regularity in repeated reasoning.

Common Core ELA:

- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases

NGSS Practices:

- 5. Using mathematics and computational thinking

Key Concepts

- Encryption is used to keep data secure as it is transmitted through the Internet.

- Symmetric encryption involves encrypting and decrypting data using an algorithm and a key.
- Encryption algorithms themselves are standardized (well known), so the key must be kept secret.

Essential Questions

- How is cybersecurity impacting the ever increasing number of Internet users?

Teacher Resources

Student computer usage for this lesson is: **optional**

In the Lesson Resources folder:

- "Cryptography Partner Practice": A worksheet for the students
- "Cipher Python Project": A worksheet with instructions for a simple Python project
- "Cipher Python Project Rubric": The rubric for the Cipher project

Optional: Lesson slides with the key questions, encryption demos, and diagrams (the teacher could simply read the questions and present demos and diagrams by writing on a board).

For examples, consider reviewing *The Code Book* by Simon Singh.

Lesson Plan

Getting Started (5 min)

Journal:

- Name one website you use that requires you to log in with a username and password.
- Why does the website require you to provide a username and password?

Introduction to Content (15 min)

Lesson Motivation [5 min]

- Present the scenario: “Alice would like to send a message to her friend Li in China, but she wants to keep it secret from everybody else.”
- Ask the students: “If Alice sends the message to Li by email over the Internet, will her message remain secret?”
- Student responses should bring up the architecture and trust model of the Internet to show that Alice’s message could be intercepted along the way, since it will pass through many devices before it ends up at Li’s computer.

Present the Key Concepts [10 min]

Tell the students, “This problem is not a new one. Throughout history, people, including government and military officials and personnel, business owners, and others, have wanted to send secret messages to someone but worried that the message could be intercepted along the way.”

There are two ways to try to keep the message secret: Steganography and Cryptography.

Explain the basic difference between the two.

- *Steganography* is when a message is "hiding in plain sight". Examples: Writing something in invisible ink that can be revealed with a special type of light.
- *Cryptography* is when a message is modified in a way that hides the meaning of the message. For example, the letters are replaced with symbols that someone else would not understand.

Present two different encryption techniques, showing one example of each.

- Transposition:
  - Definition: Rearranging the letters in the message.
  - Example: Rail fence (or "box cipher") http://www.braingle.com/brainteasers/codes/railfence.php (http://www.braingle.com/brainteasers/codes/railfence.php) or http://practicalcryptography.com/ciphers/rail-fence-cipher/ (http://practicalcryptography.com/ciphers/rail-fence-cipher/)
- Substitution:
  - Definition: Each letter is replaced by a different letter or symbol (although, technically a letter could be replaced by itself).
  - Example: Caesar Cipher (Shift Cipher) http://www.braingle.com/brainteasers/codes/caesar.php (http://www.braingle.com/brainteasers/codes/caesar.php) or http://practicalcryptography.com/ciphers/classical-era/caesar/ (http://practicalcryptography.com/ciphers/classical-era/caesar/)
  - Online Tool for Demonstration: Cipher Disk - http://inventwithpython.com/cipherwheel/ (http://inventwithpython.com/cipherwheel/)

An alternative to this lecture portion above is to have students independently study the same concepts using a reading, video, or online learning tool. Here are some suggested resources:

- Khan Academy: Have students watch the videos on "What is cryptography?" and "The Caesar Cipher" and complete the "Caesar Cipher Exploration". https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/intro-to-cryptography (https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/intro-to-cryptography)
- Practical Cryptography: Have students use the tools and descriptions to learn how to perform the rail fence encryption and decryption: http://crypto.interactive-maths.com/rail-fence-cipher.html (http://crypto.interactive-maths.com/rail-fence-cipher.html)

Summarize with this overview: "Each encryption scheme involves an algorithm and a key. The algorithm is the set of steps that you follow to accomplish the encryption. The key is the secret piece of information that is needed to know exactly how to apply the algorithm in this case. This allows you to securely send encoded information across the Internet and decode it when it arrives. Some codes are more secure than others."

Guided Activities (25 min)

Practice [15 min]

Have the students pair up and practice sending each other encrypted messages, then decrypting them to make sure they end up with the correct message.

A worksheet called "Cryptography Partner Practice" is provided in the Lesson Resources folder.

1. Each student gets to write two short messages that they will encrypt and send to their partner.
2. First message: Transposition: Use the rail fence algorithm. You must agree on the number of rails to use (this will be the “key”).
3. Second message: Substitution: Use the shift substitution cipher algorithm. You must agree on the amount to shift (this will be the “key”).
4. For each message, pass it to your partner and have them decrypt it using the agreed upon algorithm and key. Have them read back the decrypted message to make sure they decrypted it correctly.

Follow Up: Analyzing the Strength of Ciphers [5 min]

Ask the students: “How difficult would it be to crack a message that was encrypted using the Caesar (shift) cipher if you didn’t know the key? How would you do it?” (Easy, try each of the 25 possible shifts.)

Present: There are two ways to increase the strength of encryption:

Option #1: Increase the number of possible keys.

A general substitution (not limiting to just a shift) dramatically increases the number of keys. The number of keys in this case is the number of permutations (different orderings) of the 26 letters in the alphabet. This can be computed by multiplying the 26 options for the first letter in the cipheralphabet, by the 25 remaining options for the 2nd letter, 24 remaining options for the 3rd letter, etc. (26! or 26 factorial).

The answer: $4.032914e \times 10^{26}$ keys (Google will calculate it for you).

This analysis makes it seem as though a substitution cipher would be unbreakable, but clever people have invented tricks (e.g., frequency analysis) that can be used so you don't have to try all of the different keys.

Option #2: Use a better algorithm.

For example, use a polyalphabetic cipher that combines multiple cipher alphabets.

(If time allows, you can have students explore other ciphers. For further study, see Khan Academy or *The Code Book* by Simon Singh.)

Follow Up: Defining Symmetric Encryption and Seeing the Abstraction in Symmetric Encryption Systems [5 min]

Present a diagram that shows high-level view of the encryption and decryption process (see *The Code Book*, p. 11).

1. Identify this as an example of abstraction. (You can ask the students to try to explain why.) Example: This is abstraction because it shows the general process of encryption and decryption using any key or algorithm. It omits the details of the specific algorithm and the type of key.
2. Tell the students, “The types of encryption you learned today are called “symmetric”. Why do you think they are called “symmetric”? (The same key is used to encrypt and decrypt. You use the algorithm to encrypt, and then reverse it to decrypt.)
3. What do you think it would mean for encryption to be asymmetric (non-symmetric)? (foreshadowing the next lesson)

Wrap Up (5 min)

Journal:

- What is the role of the algorithm in the encryption process? What is the role of the key?
- Which one of these, the algorithm or the key, is more important to keep secret? Why?

Optional Project for additional Python Practice

Use the "Cipher Python Project" worksheet in the Lesson Resources folder. Students are tasked to create a simple Caesar cipher program that uses ASCII values to shift messages by a certain letter. The rubric for this project is also in the Lesson Resources folder.

Homework (Optional): *Choose one of the following or let each student choose which one to complete.*

1. Computer Encryption: Use bitwise XOR to do substitution cipher (see *The Code Book,* p. 247)
2. Students read a historical account that involves encryption (Mary, Queen of Scots) http://www.nationalarchives.gov.uk/spies/ciphers/mary/ (http://www.nationalarchives.gov.uk/spies/ciphers/mary/) (After reading the introduction, click on the links below the picture for “Mary’s ciphers” and “The Babington Plot”)
3. Students read about cryptanalysis and learn about the frequency analysis technique. Try using it on an encryption puzzle..
    1. An example of breaking a substitution cipher: http://www-math.ucdenver.edu/~wcherowi/courses/m5410/exsubcip.html (http://www-math.ucdenver.edu/~wcherowi/courses/m5410/exsubcip.html)
    2. Try deciphering an encrypted message using the techniques you read about: http://cryptogram.org/solve_cipher.html#contents (http://cryptogram.org/solve_cipher.html#contents)

Evidence of Learning

Formative Assessment

The teacher will evaluate student responses to the journal entries, class discussion questions, and the students performance during the encryption practice.

Internal Use Only

Pending Tasks

Create lesson slides (in progress)

the Code Book by Simon Singh is used as a reference for illustrations, I guess these will appear in the slides revamped?

(http://csmatters.org) 3 - 12

0b11 - 0b1100

Cryptography: Public Key Encryption, Certificate Authorities, and Open Standards

Unit 3. Information and the Internet

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary

In this lesson, students will learn two solutions to the key distribution problem and the mathematical foundations behind these solutions. They will make connections between encryption, the use of SSL/TLS in web browsers, and the use of digital certificates. Students will recognize the value of open standards used in modern cryptography.

Outcomes

- Students will understand the impact of the key distribution problem on secure communication.
- Students will understand that a carefully designed one-way mathematical function allows people to exchange keys or use public keys to solve the key distribution problem.

- Students will understand that digital certificates are used for authentication, and that these certificates rely on the trust model: the certificate authorities are being *trusted* to provide accurate information.

Overview

Session 1

1. Getting Started (5 min) - Students journal on the safety of transactions made online and the role of cryptography,
2. Introduction to Content (15 min) - The Key Distribution problem and mathematical modulus are presented and discussed.
3. Guided Activities (25 min) - Students learn about double encryption through analogies and a group activity.
4. Wrap Up (5 min) - Journaling on sharing secrets.

Session 2

1. Getting Started (5 min) - Students journal on how Diffie's solution benefits them.
2. Introduction to Content (5 min) - Students discuss Diffie's solution to the Key Distribution problem
3. Guided Activities (35 min) - Students Role Play and Journal about Public Key Encryption.
4. Wrap Up (5 min) - Journaling about the positive/negative aspects of Public Key Encryption

Learning Objectives

CSP Objectives

- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4B - Effective collaborative teams consider the use of online collaborative tools.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
  - 4.1.1H - Different algorithms can be developed to solve the same problem.
- 6.3.1 - Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it.
  - 6.3.1H - Cryptography is essential to many models of cybersecurity.
  - 6.3.1I - Cryptography has a mathematical foundation.
  - 6.3.1J - Open standards help ensure cryptography is secure.
  - 6.3.1K - Symmetric encryption is a method of encryption involving one key for encryption and decryption.
  - 6.3.1L - Public key encryption, which is not symmetric, is an encryption method that is widely used because of the functionality it provides.
  - 6.3.1M - Certificate authorities (CAs) issue digital certificates that validate the ownership of encrypted keys used in secured communications and are based on a trust model.

Essential Questions

- How are programs developed to help people, organizations or society solve problems?
- How is cybersecurity impacting the ever increasing number of Internet users?

Teacher Resources

Student computer usage for this lesson is: **optional**

In the Lesson Resources folder:

- Lesson slides
- Worksheet for Key Exchange Activity
- Script for Public Key Encryption 3 Act Play

Lesson Plan

Session 1

Getting Started (5 min)

Students should answer the following questions in their journals:

- What types of online activities require information to be kept secret when it is transmitted?
- How does cryptography allow for information to be kept secret when it is transmitted?

Introduction to Content (15 min)

Suggested Review

- Have students present their solutions to the "Computer Encryption" homework from Lesson 2-12 and/or connect back to the previous lesson by posing the question: "Do you think computers actually use the encryption algorithms we learned last class?"
    - This discussion should lead to the question of what algorithms computers actually use. In the 1970s, the US government chose DES as the standard encryption algorithm that everybody could use. It has been updated to AES, which is a stronger algorithm, but DES and AES are very similar.
    - The teacher should make the point that at a high level, DES/AES are symmetric encryption algorithms that work on the same basic principle of the simple algorithms we looked at in the previous lesson.
    - The teacher can leave it at that, or optionally present some brief details on DES/AES. One option is to show the first minute or two of this visualization video and talk over it: http://youtu.be/mlzxpkdXP58 (http://youtu.be/mlzxpkdXP58)

Motivation: Present the Key Distribution Problem:

Introduce the following topic. Allow for discussion among the class about possible solutions to the problem presented.

- Alice wants to send Li some secret information over the Internet. We know that she can encrypt the information before sending it, but how will Li know what key Alice used to encrypt the message?
- This is called the **Key Distribution Problem** and it has been around as long as encryption has. A whole business was developed around this challenge: people were paid to go around the world delivering briefcases full of encryption keys. This distribution process obviously can be very expensive and is completely unpractical for the average person. For a long time, nobody thought that the key distribution problem could be solved algorithmically.

After the class has come up with some ideas, reveal a solution to the problem that was found using math.

Present Key Information: Dreamers to the Rescue – Two men, two solutions, one important mathematical idea.

- Despite everybody telling them they were crazy and hopeless, Martin Hellman and Whitfield Diffie teamed up to try to solve the Key Distribution Problem. Amazingly, they each came up with a solution, both of which can be used to solve the problem.
- **Introduce One-Way Functions**: Both solutions use a mathematical concept called “one-way functions”. Most functions we are familiar with are “two-way": that is, they can easily be applied in either a forward or a reverse direction. For example, if f(x) = 2x, then it is easy to see that f(5) would be 2(5) or 10. It is also easy to see that if f(x) = 10, then 2x = 10, so x must be 5. One-way functions are different in that they are easy to use in one direction, but are very hard to reverse.
- Key Distribution – Solution #1 – Hellman’s Idea – Key Exchange Protocol

Guided Activities (25 min)

Analogy (10 min)

1. To help with the explanation of the topic, start with an analogy that uses different colors of paint.
    - Show video https://www.youtube.com/watch?v=YEBfamv-_do (https://www.youtube.com/watch?v=YEBfamv-_do) [2:25 - 4:20]
    - Alternatively, use paint to demonstrate the process live in the classroom!
2. The real system relies on a mathematical operation called *modulus* (or *clock arithmetic*), which is when you divide two numbers and find the remainder.
    - Use a clock to visually demonstrate the operation. (The number mod 12 gives you the time)
    - Show how to do modulus by using either long division, a calculator, or Google. (Consider this: If you know the number you divided by, and you know the remainder, can you easily figure out what the original number was?)
    - Alternatively, show the next section of the video: https://www.youtube.com/watch?v=YEBfamv-_do (https://www.youtube.com/watch?v=YEBfamv-_do) [4:20 - 6:18]

The system also uses powers (base/exponent). Very briefly review power notation.

Group Activity (15 min)

1. Students are paired up, and then join another pair to form a group of four.
2. Using either a set of clear directions or an online widget, one pair performs the key exchange using the $Y^x \pmod P$ expression, while the other pair listens in. They then switch roles. They should see that the key is established while sharing information publicly, but the key itself is kept secret.

Wrap Up (5 min)

Students should answer this question in their journals:

- How can two people establish a shared secret in public?

Suggested Homework:

Research Hellman and Diffie’s work on public-key exchange, identify the big ideas of CS Principles that show up, and provide specific examples of how they are related to what you find out about Hellman and Diffie's work. Alternately, read about the British group that developed the same solution as Hellman and Diffie's to public key encryption in secrecy (http://cryptome.org/ukpk-alt.htm (http://cryptome.org/ukpk-alt.htm)).

Session 2

Getting Started (5 min)

Students will read the following question and record their thoughts in their journals:

- Whitfield Diffie said that he wanted to solve the key distribution problem for benefit of "ordinary people," as opposed to just governments and corporations. How do you and I benefit from his team's solutions to the Key Distribution Problem?

Have students present ideas from their journal entries. Use this as a way to review the Key Distribution Problem, and the team that tackled the problem.

Introduction to Content (5 min)

Present Diffie's Solution - Public Key Cryptography

- **The Idea**: Encrypt with one key (public key), decrypt with a second key (private key)
- For further clarification, use this analogy:
  - Analogy with Physical Locks – Person B gives out open padlocks (public key) to anybody who wants to send him or her something secret. Person A just puts the secret in a box, and shuts the padlock (easy to do!). When Person B receives the box, they use the combination (private key) to unlock it.
- This is called "**Asymmetric Encryption**" since it uses two different keys.
- Consider what happens when you go to a "secure" website to check out when you are finished shopping at an online store. Your browser says "https" and some show a picture of a lock. The system, called SSL (Secure Sockets Layer) or the updated version TLS (Transport Layer Security) takes advantage of Diffie's system.

Guided Activities (35 min)

Part 1 (20 min) - Role Play

Have students act out three short scenes (see "Public Key Encryption Plays") in order to illustrate how the system works. (It is advisable to select "dramatic" students to fill the four roles.)

Roles

- Customer
- Store
- Store Impersonator
- Certificate Authority

Overview

- Act 1: The customer and store use public key encryption to complete an online purchase using a credit card. (All seems well, but the next act will have a twist!)
- Act 2: The store impersonator distracts the store and jumps in, steals the credit card info. (After this, stop for a minute and ask the students to explain what the problem is)
- Act 3: Repeat Act 2, except that now the customer asks the Certificate Authority to verify the public key. The impersonator is revealed as a fraud, then the real store completes the transaction with the CA verifying.

Follow Up question to ask the students: Who do you have to trust for this system to work? (2 min)

- Sample Answer: The Certificate Authority. You are trusting that they are giving you valid information so you can verify the identity of others. (A student may ask how you know the certificate authority is not being impersonated. The answer is that the public key of the certificate authority is saved in your browser so you can verify their identify yourself.)

Optional Section on Mathematical Foundation

What are the mathematical details that enable this idea of work? (Don't worry, we are not going to fully answer this!)

- Diffie didn't actually figure out the math to make this actually work, he just had the key idea (Example of Abstraction!) He put the idea out there for others to figure out the details of the math that would make it work (Example of Collaboration!)
- One system (RSA) multiplies prime numbers as part of the one way function. It is easy to multiply two prime numbers, but it is very hard to determine the factors of the product (if you didn't already know them).
  - Give the students a couple of problems to illustrate this.
  - The two primes that you multiply are essentially the "private key". The product is the "public key".
  - This works in practice because the numbers used are huge, making the factoring process extremely difficult and time consuming, even with a large amount of computational power.

Discussion: Do "Open Standards" make sense in the world of Cryptography?

The systems of encryption used on the web have been "standardized" (meaning that everyone agrees to use the same systems) so that computers all over the world can communicate with each other. These standardized systems could be "proprietary"(meaning the details are kept secret), or they can be "open" (meaning the details are shared for anybody to see).

Part 2 (15 min) - Think-Pair-Share

Students will Think-Pair-Share about the following prompt:

- If cryptography is all about secrecy, then does it make sense to have open standards of encryption? List all the pros and cons that you can think of for open standards.

Possible responses

Benefits

- People can independently verify that the algorithm is strong, secure, and doesn't have vulnerabilities.
- People can make sure there are no "back doors" in the algorithm that let certain people spy on them even without the key.

Note: Heartbleed vulnerability is a good example of something that was eventually caught because of open standards. (This could be a homework assignment to read about it)

Drawbacks

- Any cyber-criminal can look at the algorithm and try to find a vulnerability to exploit.
- People may assume that because it is open, all the vulnerabilities have been found and plugged when that is not necessarily true.

Wrap Up (5 min)

Students should read this question and record their thoughts in their journals:

- "Open standards result in strong security". Do you agree or disagree with this statement? Give specific reasons to back up your position.

Optional Homework:

Read about Heartbleed vulnerability in SSL. Reflect on how open standards relate to this.

Extensions:

RSA Encryption Algorithm Video: http://youtu.be/M7kEpw1tn50 (http://youtu.be/M7kEpw1tn50)

Evidence of Learning

Formative Assessment

The teacher will observe and evaluate student responses to journal entries, class discussion questions, and class activities.

Internal Use Only

Future Work

Create online widget for Key Exchange Activity.

(http://csmatters.org) 3 - 13

0b11 - 0b1101

Optional

Cybersecurity: Malicious Code, Identity Theft, and Remedies

Unit 3. Information and the Internet

Revision Date: Aug 20, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

This lesson will increase student awareness of the concept that there are dangers associated with Internet usage. It addresses Internet Security with issues inherent to Internet usage: viruses, worms, Trojan horses, and identity theft. The primary objective of this lesson is to equip students with knowledge that will enable them to make responsible choices regarding their Internet use, to prevent security risks.

Outcomes

- Students will be able to identify key general attributes of the threats to the security of computers and information via the Internet such as viruses, worms, and Trojan Horses.
- Students will understand critical attributes of the sources, and consequences to individuals and society, of identity theft.
- Students will understand how to protect themselves and their computers from external threats.
- Students will develop a strategy to inform others of the security risks inherent to Internet usage.

Overview

Session 1

1. Getting Started (10 min) - Introduce vocabulary and discuss Internet security.
2. Guided Activity (35 min) - Students explore malicious code, identity theft and its effects
3. Wrap Up (5 min) - Journaling about problems associated with the Internet.

Learning Objectives

CSP Objectives

- 6.1.1 - Explain the abstractions in the Internet and how the Internet functions.
  - 6.1.1A - The Internet connects devices and networks all over the world.
- 6.3.1 - Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it.
  - 6.3.1C - Implementing cybersecurity has software, hardware, and human components.
  - 6.3.1D - Cyberwarfare and cybercrime have widespread and potentially devastating effects.
  - 6.3.1E - Distributed denial-of-service attacks (DDoS) compromise a target by flooding it with requests from multiple systems.
  - 6.3.1F - Phishing, viruses, and other attacks have human and software components.
  - 6.3.1G - Antivirus software and firewalls can help prevent unauthorized access to private data.

Students will:

• Identify key general attributes of the threats to the security of computers and information via the Internet such as viruses, worms, and Trojan Horses.

• Understand critical attributes of the sources, and consequences to individuals and society, of identity theft.

• Understand how to protect themselves and their computers from external threats.

• Develop a strategy to inform others of the security risks inherent to Internet usage.

Key Concepts

Students will:

- Learn about the different types of malicious code and how to take prevention steps to safeguard systems, data, and identity.
- Give advice on secure cyber practices.

Essential Questions

- How is cybersecurity impacting the ever increasing number of Internet users?
- What are some potential beneficial and harmful effects of computing?
- What are different types of malicious code and what is the intention of each attack?
- How can internet users protect themselves from malicious code and prevent such cybercrime attacks?
- How can internet users follow secure practices to reduce the risk of identity theft?

Teacher Resources

Student computer usage for this lesson is: **required**

Teacher's resources:

- NA SAIT Security Video - Malicious Code - Malware Video - https://www.youtube.com/watch?v=7wAHZLFiY-E (https://www.youtube.com/watch?v=7wAHZLFiY-E)
- What to do if you are a victim of Identity Theft: ***Possible Answers*** -
  - The FTC's website is a one-stop resource to both learn about identity theft and walk you through the appropriate actions if your identity is stolen: http://www.ftc.gov/bcp/edu/microsites/idtheft/ (http://www.ftc.gov/bcp/edu/microsites/idtheft/)
  - Some possible steps if your identity is stolen:
    - Report the identity theft to the three major credit bureaus: Experian, TransUnion, and Equifax.
    - File a police report with local law enforcement.
    - Report the theft to the FTC online at www.ftc.gov/idtheft or by phoning 1-877-ID-THEFT (1-877-438-4338).
  - Possible ways for Deterring Identity Theft:
    - Shred financial documents that are not being kept for safeguarding. [This allows a teacher to cover the kind of information that should be held and for how long (in years). It also allows a teacher to cover what documents are best kept in a safe deposit box, a home safe, regular home files, etc.]
    - Do not carry around your Social Security card in your wallet.
    - Do not give out personal information over the phone or over the Internet unless you are absolutely sure who you are dealing with.
    - Choose computer and electronic passwords with care by avoiding birth dates, your Social Security number, your mother's last name, etc.
    - Try not to have your postal mail pile up in your mailbox for several days; if you are going to be away for a few days, have your mail held at the post office until you return.
    - Do not click on suspicious links in e-mail or complete forms with your account number and password. Check the web address.
    - Be suspicious about regular bills that do not arrive on time, denials of credit for no apparent reason, calls or letters about purchases you did not make, charges on your financial statements that you do not recognize.
    - Use a password to access your mobile devices such as your cell phone, tablet (iPad), etc., just as you would have a password to get access to your e-mail accounts.

Students' resources:

- writing journals
- blogs

Lesson Plan

Session 1

Getting Started (10 min)

1. **Journal:** How might malicious code be a threat to Internet security?
   - Possible answers to bring up: identity theft, data theft, stealing money, breaking down other computers, turning other computers into bots to create more problems, etc.
2. **Introduce the topic/Discussion**: Inform the students that today they will be talking about Internet security and participating in discussions about the dangers of viruses and , identity theft and malware along with resources and prevention tips.
   - Malicious code is programming code designed with a harmful intent (to hack, cause damage, etc.). With Internet usage comes rights and responsibilities to protect your computer from malicious code. Malicious code causes millions of dollars in damage every year.
   - How can malicious code spread across many computers so quickly?
   - Examine the idea of interconnectedness.

3. **Introduce key vocabulary - have students crowdsource definitions or look up separately and compare.**
   - Computer virus
   - Spyware
   - Ransomware
   - Bot
   - Hacker
   - Malware
   - Adware

Guided Activity (30 min) - Malicious Code and Identity Theft

1. Play the NA SAIT Security Video: "Malicious Code - Malware" at https://www.youtube.com/watch?v=7wAHZLFiY-E (https://www.youtube.com/watch?v=7wAHZLFiY-E) (3 minutes, from 2013)
   - AnnMarie Keim, IT Specialist, discuss the concepts of Internet Security and introduce the different types of malicious code and how to protect from this type of cybercrime. Keep your system updated, use antivirus and firewall. Backup important files.

2. Review the following words from previous lessons:
    - routers
    - firewalls
3. Discuss: Is malcious code the only way to cause harm on the Internet? (no)
    - https://vimeo.com/63422786 (https://vimeo.com/63422786) (0:30) Google mini-video to show what phishing is, using a brick through a glass window example to get attention.
    - Talk about social hacking, using ordinary everyday life tricks to get information like looking over somebody's shoulder when they type their password, sneaking inside of a locked door before it closes, etc. Point out that it is a combination of people, software and hardware that are both the problem and the solution to Internet security.
4. Students work in groups to explore selected topics. See the document in the lesson folder: Lesson 13 Cybersecurity: Malicious Code, Identity Theft Additional resources for possible resources. Or search and look for a reliable provider of information. Students create a blog post about their topic to share at the end of class.
    - Suggested topics: (decribe how they work and what can be done to protect against them in terms of human behavior, software and hardware)
        1. Famous viruses,
        2. Virus Hoaxes
        3. Worms
        4. Trojan Horses
        5. identity theft
        6. Phishing scams
        7. Cellphone and texting scams
        8. Ransomware
        9. Firewalls
        10. Antivirus and anti-malware protection

Wrap Up (5 min)

Students will read the following prompt and respond in their journals:

- The only 100% way to prevent malicious code attacks and identity theft is to not go on the Internet. Do you see that as a viable solution for individuals? Corporations? Support your answer.

Consider the following questions and discuss answers as a class:

1. Have you or someone you've known experienced a virus, or malware? What was the outcome? What did you take away from this experience?
    - Cover the following:
        - Time and Money spent – (by corporations and by individual to protect computer)
        - What were the consequences?
        - What did the victim go through?
2. How can you avoid malicious code?
    - *Possible Answers*:
        - Anti-virus software
        - Anti-spyware software
        - Anti-adware software
        - Restore points
        - Keep patches and updates current on your computer
        - Careful use of email
        - Careful use when downloading items

Homework (optional)

On your home computer, see how vulnerable you are to malware and identity theft:

- Carry out some remedies and prevention tips (minimum of three tasks) that you learned today.
- On your blog, list what you did to safeguard your system, your data, and your identity.
- Be prepared to share in the next class

Options for Differentiated Instruction

Extensions/Differentiation:

- Malicious Activity World Map - http://www.team-cymru.org/Monitoring/Malevolence/maps.html (http://www.team-cymru.org/Monitoring/Malevolence/maps.html) - a map that shows malicious activity in world
- Malware Classifications - http://maple.cs.umbc.edu/ce21/instruction/login (http://maple.cs.umbc.edu/ce21/instruction/login) - provides classifications of different types of malware

- Current Threat Activity - http://www.trendmicro.com/us/security-intelligence/current-threat-activity/ (http://www.trendmicro.com/us/security-intelligence/current-threat-activity/) - shows what the current malware threats are and what can be used to prevent them from impacting your computer.
- Threat Encyclopedia - http://about-threats.trendmicro.com/us/glossary/all (http://about-threats.trendmicro.com/us/glossary/all)
- Threat Intelligence Resources - http://about-threats.trendmicro.com/us/infographics (http://about-threats.trendmicro.com/us/infographics)
- Why do cyber attackers do what they do? - http://about-threats.trendmicro.com/us/security-roundup/2014/1Q/cybercrime-hits-the-unexpected/ (http://about-threats.trendmicro.com/us/security-roundup/2014/1Q/cybercrime-hits-the-unexpected/)

If there is additional time, watch one of the TED talks

- James Lyne: Everyday cybercrime -- and what you can do about it (17:26)
- The Internet is on fire | Mikko Hypponen | TEDxBrussels (19:16)

Students can create an "Identity Theft Prevention Action Plan," including a purpose and list of ten guidelines, to share with family and friends after they have researched prevention tips on the FTC website.

- Students may use their choice of the following Web 2.0 websites to create their action plan:
  - http://www.powtoon.com/ (http://www.powtoon.com/) - create a presentation
  - http://www.livebinders.com/welcome/home (http://www.livebinders.com/welcome/home) - create a curation
  - http://goanimate.com/ (http://goanimate.com/) - create an animation
  - https://www.podomatic.com/login (https://www.podomatic.com/login) - create a podcast
  - http://piktochart.com/ (http://piktochart.com/) - create an infographic
- Students will embed their action plan to a blog post in order to share with teacher, classmates, friends, and family. Students should email friends and family (minimum of three people with the teacher cc'd) the link to this blog post on Identity Theft Prevention Action Plan (title of blog post and subject line of email).

Evidence of Learning

Formative Assessment

- Journal writings
  - Introduction question prompts
  - Wrap-up question prompt
- Class discussions - answers, input, and further inquiry by students
- Identity theft prevention plan

Summative Assessment

Unit Assessment and Investigate/Explore Performance Project – at end of unit.

(http://csmatters.org) 3 - 14

0b11 - 0b1110

Unit 3 Assessment

Unit 3. Information and the Internet

Revision Date: Jul 22, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

A general review of the essential vocabulary and concepts covered in Unit 2.

Learning Objectives

Not only do students need to review the concepts covered in the unit to prepare for the written questions at the end of the course, but also, the teacher needs time to review and give feedback on the practice Explore task before assigning the actual AP graded Explore task.

Teacher Resources

Lesson Plan

(http://csmatters.org) E - 1

0bE - 0b1

Explore - Impact of Computing Innovations

Unit Explore Performance Task

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 10 50-minute sessions

Lesson Summary

Summary

In the Explore Performance Task (EPT), students choose and explore a computing innovation. The EPT requires students to select and investigate a computational innovation that:

- Has or has had the potential to have significant beneficial and harmful effects on our society, economy, or culture.
- Consumes, produces, and/or transforms data.
- Raises at least one data storage concern, data privacy concern, or data security concern.

Students are expected to complete the EPT with minimal assistance from anyone. Students will have 8 hours of class time to complete, and submit:

- Computational Artifact
- Written Responses

This lesson plan provides a schedule for 10 50-minute sessions, in order to meet the 8-hour in-class minimum required by the College Board.

Overview

Session 1: Identify performance task requirements and choose a computing innovation

Session 2: Review EPT Rubric and Research - Day 1

Session 3: Draft Written Responses part 2A, 2C, and 2E

Session 4: Research - Day 2

Session 5: Artifact Design

Session 6: Artifact Development

Session 7: Artifact Development

Session 8: Written Responses to parts 2B, 2D, and 2E

Session 9: Complete Artifact and Written Responses and Create PDF versions

Session 10: Submit Artifact and Written Responses

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
  - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
  - 1.2.2B - A creative development process for creating computational artifacts can be used to solve problems when traditional or prescribed computing techniques are not effective.
- 3.1.3 - Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
  - 7.1.1L - Computing contributes to many assistive technologies that enhance human capabilities.
  - 7.1.1M - The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
- 7.5.1 - Access, manage, and attribute information using effective strategies.
  - 7.5.1A - Online databases and libraries catalog and house secondary and some primary sources.
  - 7.5.1B - Advance search tools, Boolean logic, and key words can refine the search focus and/or limit search results based on a variety of factors (e.g., data, peer-review status, type of publication).
  - 7.5.1C - Plagiarism is a serious offense that occurs when a person presents another's ideas or words as his or her own. Plagiarism may be avoided by accurately acknowledging sources.
- 7.5.2 - Evaluate online and print sources for appropriateness and credibility.
  - 7.5.2A - Determining the credibility of a source requires considering and evaluating the reputation and credentials of the author(s), publisher(s), site owner(s), and/or sponsor(s).
  - 7.5.2B - Information from a source is considered relevant when it supports an appropriate claim or the purpose of the investigation.

7.3.1 A

Key Concepts

Students will be able to identify beneficial and harmful effects of a computing innovation.

Students will be able to explain how the innovation uses, produces, and/or transforms data.

Students will be able to explain a privacy, security, or data storage concern related to the innovation.

Teacher Resources

Student computer usage for this lesson is: **required**

AP Computer Science Principles Explore Performance Task -- Explore_Performance_Task_Nov2015.pdf in lesson resources folder

AP Computer Science Principles Performance Task Rubric -- Explore_Performance_Task_Rubric_Nov2015.pdf in lesson resources folder

Lesson Plan

Teacher Guidelines

Prior to Administration of the Performance Task

Teachers should have instructed students in each of the following before this lesson.

- How to obtain reliable sources to support facts
- Use of organizational and prewriting tools (http://slc.berkeley.edu/you-start-writing-paper-guide-prewriting-techniques-0)
- Use of computational tools needed for creation of computational artifacts
- Proper citations of references
- How to distinguish between effective and ineffective artifacts
  - Samples may be used
  - Effective artifacts help the reader/viewer understand the beneficial and harmful impacts of the innovation
  - Ineffective artifacts repeat information in the written responses, are multi-slide presentation with paragraphs or bullets, are not original - copy and paste - or emphasize the functionality of the innovation rather than its impact
- How to distinguish computing innovations from other innovations

Teachers and classmates may help students to understand the Explore Performance Task, help students manage the process, and help them to submit the required projects. Once the work on the project begins, however, students must work individually without assistance.

At the beginning this Explore Performance Task Lesson:

Teachers should influence student selection of a computing innovation (see Exploring Computing Technologies in the Explore_Performance_Task_Nov2015 in lesson folder). However, teachers may not require selection of particular topics for students.

Direct students to find a computing innovation and its:

- Beneficial and harmful impacts on society, economy, and culture
- Use, production, and/or transformation of data
- Relationship to a concern about data storage, privacy, or concern

Guide student understanding of the products that they must produce, including:

- The nature of each product
- The level of detail required
- The time that students have to complete and submit the projects

During the Performance Task

Teachers should:

- Resolve hardware/technical problems.
- Manage student progress throughout the task.

Teachers may:

- Perform summative assessment; however, may not provide any feedback.
- Clarify directions for the performance tasks.
- Remind students of submission requirements.

Teachers may not:

- Provide or distribute the topics that students have chosen to other students.
- Write, revise, or amend student work.
- Perform research for students.
- Provide boilerplates.
- Allow students to submit practice tasks as actual performance tasks.
- Suggest or evaluate student products until after the computational artifact and written responses are submitted.

Overview

Session 1 develops student understanding of the EPT requirements and guides selection of a computing innovation to explore. If this was done in the practice task, then questions addressed in this session may be assigned as homework, with students using the first day to do preliminary research.

Session 2 develops student understanding of the rubric, so they know the level of expectations that readers will have when scoring them. If this was done during the practice EPT, then the questions assigned in the section may be assigned as homework and the full day used for student research.

Session 1: Identify performance task requirements and choose a computing innovation.

Introduce the Explore Performance Task (EPT)

Tell students:

Today you will begin the Explore Performance Task. It is an exploration of a computing innovation of your choice according to the guidelines established by the College Board. The goal of this task is to deepen your understanding of computer science principles.

Exercise 1:

Provide students with a copy of the Explore Performance Task (Explore_Performance_Task_Nov2015 in lesson folder) description from the College Board. Students should read the Overview section on page 1 and answer the three questions below, sharing their answers with elbow partners. (2 min)

1. Who chooses the computing innovation you will explore?
2. What two products are you to produce?
3. How much time will you have to complete and submit the required products?

After students share with elbow partners, address any questions students have to this point. (3 min)

Students should read the General Requirements section beginning on page 1 and answer the seven questions below sharing their results with elbow partners. (10 min)

1. What three aspects of the computing innovation will you investigate?
2. What are three types of references that you may use for the project?
3. How many sources are you required to use?
4. How many of the sources must either be in print or online?
5. How many of the sources must have been created since the end of the previous academic year?
6. What must the computing artifact that you produce represent or explain?
7. What do your written responses need to address?

After students share with elbow partners, address any questions that students have to this point. (5 min)

Students should read just the first two paragraphs of the Submission Requirements section on page 2 and answer the four questions below sharing their results with elbow partners. (3 min)

1. What must your computational artifact explain or illustrate?
2. What must the computational artifact not do?
3. What is the computational artifact's maximum size?
4. Your written response is a document with five sections to be labeled 2a - 2e in order. What is the total number of words that your written response may contain?

After students share with elbow partners, address any questions students have to this point. (2 min)

Assign students to 5 groups and assign a section a - e to each group. Have each group answer these two questions for each section (10 min)

1. What topics are to be addressed in section 2a?
2. What is the suggested length (number of words) for this section?
3. What topics are to be addressed in section 2b?
4. What is the suggested length (number of words) for this section and how detailed should it be?
5. What topics are to be addressed in section 2c?
6. What is the suggested length (number of words) for this section?
7. What topics are to be addressed in section 2d?
8. What is the suggested length (number of words) for this section?
9. What is to be included in section 2e?
10. How are the items in this section to be cited in the written response?

After completing the four exercises, students are to choose a computing innovation to explore and share their choice with you. (15 min)

Before the next lesson, students must select and submit to you the name of the computing innovation.

Session 2: Research day 1

Tell students:

Today will be your first research day. Before starting your research, we will examine the rubric that readers will use when scoring your computational artifact and written response.

Provide students with a copy of the Explore Performance Task Rubric (Explore_Performance_Task_Rubric_Nov2015.pdf in the lesson folder). Jigsaw the rubric. Assign students into five groups and assign one row of the rubric to each group. Students are to prepare a 60-second response to these three questions about their row of the rubric. (2 min)

1. What is the content area?
2. How does a medium-quality response or artifact differ from a poor one?
3. How does a top-rated response or artifact differ from a medium one?

Each student group shares their insights based on these three questions. (5 min) Students put any questions that they have about the rubric on a post it note and put the question on the board.

Students must work individually from this point until the projects are ready to be submitted.

Students research their computing innovation, especially addressing the first two components of written response 2A and written response 2C, while collecting references to be included as a part of written response 2E.

At the end of class, students complete a daily progress report. Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content.

Session 3: Draft Written Responses to part 2A, 2C and 2E

Tell students:

> Today, students are to plan and draft responses to the first two parts of written response 2A and written response 2C, and begin creating reference section 2E. Students should include citations for information in parts 2A and 2C.

Respond to each of the student's questions posted on the board. You may clarify student understanding of project requirements and expectations.

After drafting these responses, students continue researching the computing innovation.

At the end of class, students complete a daily progress report. Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content.

Session 4: Research Day 2

Tell students:

Students are to continue research into their computing innovation focusing on written response 2D while collecting references to be included as part of written response 2E.

At the end of class, students complete a daily progress report. Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content.

Session 5: Artifact Design

Tell students:

Today, you will begin the second portion of the EPT by planning your Computational Artifact.

Students reread the Explore Task Submission Requirement item 1 and answer the following 2 questions. (2 min)

1. Which of the following will your computational artifact address about your selected computing innovation?
    A. intended purpose
    B. its function
    C. its effect
2. What will your computational artifact illustrate or explain beyond what is in your written responses so far?

Plan to accomplish the goals you specified in your answers to question 1 and 2.

- Describe the artifact that you will produce.
- List the computing tools that you will use.
- List the computing techniques that you will use.
- Layout the artifact that you will create

At the end of class, students complete a daily progress report. Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content.

Session 6: Artifact Development

Tell students:

> This is the first of two days scheduled for students to create their computational artifacts. Start today by planning what you need to get done each of the two days and how much will be done on this first day.

After planning the two days of artifact development, students complete today's portion.

At the end of class, students complete a daily progress report. Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content.

Session 7: Artifact Development

Tell students:

> This is the second of two days scheduled for students to create their computational artifacts. After this session the next two sessions are intended for students to complete both written responses and the computational artifact.

At the end of class, students complete a daily progress report. Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content.

Session 8: Complete Written Responses to parts 2B, 2D and 2E.

Tell students:

> This is the first of two days scheduled for students to complete the EPT. This session is scheduled to focus on the written response. As time permits students may work on their computational artifact.

Students complete their written response.

At the end of class, students complete a daily progress report. Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content.

Session 9: Complete Artifact or Written Responses and Create PDF versions

Tell students:

This session is for students to complete constructing the two required documents of the EPT and save them in an appropriate format.

Reread and revise each Written Response including an inspection of references, associated citations and total word limits. Save the written response in PDF format.

If students are submitting their computational artifact as a PDF, it may not exceed 3 pages. The College Board also accepts multimedia files containing the computational artifact. Acceptable formats are mp3, mp4, wmv, avi, mov, wav or aif. If students elect to create multimedia files, the files may not be longer than 1 minute nor larger than 30MB in size.

At the end of class, students complete a daily progress report. Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content.

Session 10: Submit Computational Artifact and Written Response

Tell students:

There are three goals for this session.

- Submit all EPT documents by uploading them to the performance task submission web site.
- Present all computational innovations to the class.
- Complete a post task reflection.

Students upload both files. (15 min)

Computational Artifact Presentations

Post Task Reflection

What did you learn about the computing innovation you explored?

What advice would you give to students in next year's class?

Options for Differentiated Instruction

Evidence of Learning

Formative Assessment

Daily planning and progress reports

Summative Assessment

Students may not recieve feedback on the content of the EPT until after it has been submitted. Once they are submitted to the College Board EPT can be assesed using the EPT rubric.

Internal Use Only

Future Work

The lesson spans 10 days of classes but this may not be enough time if teacher uses the bulk of the first two days to guide students through the start of the project. Hopefully, in future years, more of this can be incorporated into the practice task leaving more time for the students to create their artifacts.

(http://csmatters.org) 4 - 1

0b100 - 0b1

Data Acquisition and Analysis

Unit 4. Data Acquisition

Revision Date: Jul 24, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary

In this lesson, students will learn how to acquire and analyze data to find answers to questions and solutions to problems. Students will consider whether or not the data they are presented with is necessarily valid, and research some of the various data sources online.

Outcome

- Students will explore how computation can be employed to help people process data and information to gain insight and knowledge.
- Students will learn how computation can be used to facilitate exploration and discovery when working with data.
- Students will consider what considerations and trade-offs arise in the computational manipulation of data.
- Students will explore opportunities that large data sets provide for solving problems and creating knowledge.

Overview

Session 1

1. Getting Started (5 min) - Students journal on the importance of validating data
2. Discuss journal prompt (5 min)
3. Brainstorm types of online data (5 min)
4. Explore how meaning is created from data (10 min)
5. Work with data online (20 min)
6. Assign homework (5 min)

Session 2

1. Getting Started (5 min) - Students journal about what they learned the previous day
2. Analyzing Data (30 min) - Discuss correlation and causation; discuss and explore different types of data and analysis
3. Present homework findings (10 min)
4. Wrap Up - Journal (5 min)

Learning Objectives

CSP Objectives

- 3.1.2 - Collaborate when processing information to gain insight and knowledge.
  - 3.1.2B - Collaboration facilitates solving computational problems by applying multiple perspectives, experiences, and skill sets.
  - 3.1.2C - Communication between participants working on data-driven problems gives rise to enhanced insights and knowledge.
  - 3.1.2F - Investigating large data sets collaboratively can lead to insight and knowledge not obtained when working alone.
- 3.2.1 - Extract information from data to discover and explain connections or trends
  - 3.2.1B - Large data sets provide opportunities for identifying trends, making connections in data, and solving problems.
  - 3.2.1D - Search tools are essential for efficiently finding information.
  - 3.2.1E - Information filtering systems are important tools for finding information and recognizing patterns in the information.
- 3.2.2 - Determine how large data sets impact the use of computational processes to discover information and knowledge.
  - 3.2.2A - Large data sets include data such as transactions, measurements, texts, sounds, images, and videos.
  - 3.2.2F - The size or scale of a system that stores data affects how that data set is used.

Students will be able to answer:

- How can computation be employed to help people process data and information to gain insight and knowledge?
- How can computation be employed to facilitate exploration and discovery when working with data?
- What considerations and tradeoffs arise in the computational manipulation of data?
- What opportunities do large data sets provide for solving problems and creating knowledge?

Key Concepts

Students will be able to acquire data and analyze it to find answers to a specific question or solutions for a specific problem.

Essential Questions

- What opportunities do large data sets provide for solving problems and creating knowledge?

Teacher Resources

Student computer usage for this lesson is: **required**

Student computer usage for second lesson is: **optional**

Teacher Resources

In the Lesson Resources Folder

- PowerPoints: "Finding Data" and "Finding and Analyzing Data"
- Session 1 Homework: "Homework Unit 4 Lesson 1"

Webpages Session 1

- Rebecca Dovi's Blog: http://supercomputerscience.blogspot.com/2013/03/data-unit-day-one.html (http://supercomputerscience.blogspot.com/2013/03/data-unit-day-one.html)
- Bytes Review: http://highscalability.com/blog/2012/9/11/how-big-is-a-petabyte-exabyte-zettabyte-or-a-yottabyte.html (http://highscalability.com/blog/2012/9/11/how-big-is-a-petabyte-exabyte-zettabyte-or-a-yottabyte.html)
- TV Resolutions: http://www.rtings.com/info/what-is-the-resolution (http://www.rtings.com/info/what-is-the-resolution)
- New 3D Projector: https://www.amctheatres.com/sony4k (https://www.amctheatres.com/sony4k)
- Wolfram Alpha: https://www.wolframalpha.com/tour/what-is-wolframalpha.html (https://www.wolframalpha.com/tour/what-is-wolframalpha.html)

Webpages Session 2

- Strange Corrolations: http://www.tylervigen.com/spurious-correlations (http://www.tylervigen.com/spurious-correlations)
- What is a Data Scientist? https://www.youtube.com/watch?v=9PIqjaXJo7M (https://www.youtube.com/watch?v=9PIqjaXJo7M)
- What does a Data Scientist Do? https://www.youtube.com/watch?v=vowXaEDh1uk (https://www.youtube.com/watch?v=vowXaEDh1uk)

Lesson Plan

Session 1

For this session, use the presentation "Finding Data" in the Lesson Resources Folder.

Getting Started (5 min)

Given this data: [slide 1]

A blood drive at the local high school reveals that 20% of the students were HIV positive.

Journal on these questions:

- What is your immediate reaction?
- What questions do you have?

Activities (40 minutes)

Activity 1 (5 min) - Discuss the journal prompt

Lead the students in discussion using the bullets below and slide 2 of the PowerPoint as guidence. Students should talk about WHY they assumed the data was true, or were uncomfortable questioning the truth of the data.

- Two common reasons:
    1. Because their teacher told them.
    2. It had a % sign, so it looked authoritative. We all do that sometimes.
- From Rebecca Dovi's Blog: http://supercomputerscience.blogspot.com/2013/03/data-unit-day-one.html (http://supercomputerscience.blogspot.com/2013/03/data-unit-day-one.html)
- This discussion should reinforce that data online is not necessarily complete, up to date, or even valid.

Activity 2 (5 min) - Brainstorm: What kinds of data can be found online?

Part 1 - Discussion

Data comes from many places and takes many forms [slide 3]

- Have students discuss: How do business, personal, government and devices create and use data?

Part 2 - Brainstorm

Brainstorm as a class: what kinds of data are generated? Possible answers:

- video: movies, webcam images, CCTV, youTube, Netflix, Facebook, etc.
- pictures: maps, Instagram, photos, cartoons, drawings, …. everything!
- words: books, articles, news, stories, blogs, Facebook
- numbers: facts, financial transactions, scientific data
- sound: music, speech
- behavior tracking: GPS, click behavior, search history

Activity 3: How is meaning created from data? (10 minutes)

1. Look at some data gathered about selfies from different cities around the world. [slide 4]
    - Main ideas:
        - You have to gather the data and analyze it to create meaning.
        - Creating meaning from pictures still takes some human interpretation.
    - Prompt students to come to a conclusion about the graphed data on the page.
    - Question for discussion: How large of a sample is needed to draw a conclusion?
2. Quick review: Make the point that there is a LOT of data even in a single picture. [slide 5]
    1. Define these and put them in order. Use this webpage to review bytes: http://highscalability.com/blog/2012/9/11/how-big-is-a-petabyte-exabyte-zettabyte-or-a-yottabyte.html (http://highscalability.com/blog/2012/9/11/how-big-is-a-petabyte-exabyte-zettabyte-or-a-yottabyte.html)
        - MB, bit, TB, ZB, byte, GB, pixel (one dot of color on the screen), KB, PB
    2. Look at the photo on slide 5.
        1. 365 gigapixels is 365 billion pixels, if the picture is a square, then it is 604,152 pixels on each side (too big to fit on any HDTV screen)
        2. http://www.rtings.com/info/what-is-the-resolution (http://www.rtings.com/info/what-is-the-resolution) A 4K super high resolution TV is only about 3,000 X 2,000 pixels. Even a movie screen can't show all of the detail!
        3. https://www.amctheatres.com/sony4k (https://www.amctheatres.com/sony4k), you can only look at it one part at a time.
    3. Preview Wolfram Alpha, an engine for providing knowledge from data.
        - Show the introductory video: https://www.wolframalpha.com/tour/what-is-wolframalpha.html (https://www.wolframalpha.com/tour/what-is-wolframalpha.html) (1:18) [slide 6]
        - Students will explore these sites in the next activity.

Activity 4 (20 min) - Work with some data online

1. Students should complete the Data Search and Analysis Handout. [slide 7]
    - Depending on how much time you have, you can pair students and assign even/odd questions or chunks of questions to different groups, or have each student research on their own.
2. If there's time in class, try to go over results and compare (especially the first 5) to see if people got similar answers. Why or why not? [slide 8]

Assign Homework (5 minutes)

Give students the worksheet: Homework Unit 4 Lesson 1.

There are 10 videos to choose from, each 10-15 minutes long. Either allow students to self-select, or assign them a particular video. Students should watch the video and answer the questions on the worksheet. This is an opportunity to discuss plagiarism: students are expected to watch the video and write from their own experience.

Session 2

For this session, use the presentation: Finding and Analyzing Data from the Lesson Resources Folder

Getting Started (5 min)

Students should journal on the following: Describe at least 2 ways that we create meaning out of data. [slide 1]

- Possible answers: graph it, total it, average it, find min and max, map it, compare it to other data, find trends, generate predictions (like weather), draw conclusions (facial recognition, emotions, voice inflection), diagnose diseases, discover new stars, etc.

Activities (40 min)

Activity 1 (35 min): Analyzing Data

Part 1: Correlation vs. Causation

1. Look at slide 2 from the PowerPoint. Creating meaning from data can be misleading.
2. Point out that the graph shows a direct relationships between the number of divorces in Maine and the amount of margarine that is purchased. When one goes up, the other does too, and vice versa. Is this a causal relationship?
    - Show some examples from the Tyler Vigen website http://www.tylervigen.com/spurious-correlations (http://www.tylervigen.com/spurious-correlations) . It has many examples of data connections that may be statistically valid but don't make sense. The site was created to point out how comparisons due to data correlation are often not valid.

Part 2: Data Science

1. What does a data scientist do? [slide 3]
    - Show the two videos and discuss.
        - What is a data scientist? (0:54) https://www.youtube.com/watch?v=9PIqjaXJo7M (https://www.youtube.com/watch?v=9PIqjaXJo7M)
        - What does a data scientist do? (0:39) https://www.youtube.com/watch?v=vowXaEDh1uk (https://www.youtube.com/watch?v=vowXaEDh1uk)
        - Data science is a relatively new field combining computer science with statistical data analysis and processing data to create meaning.
    - Tricks to analyzing big data:
        - Knowing what data to use, and what to disregard.
        - Knowing how to make up for missing data.
        - Knowing how to discover and predict trends and correlations.
    - There are many degrees offered in data science, and free online courses are available from Udacity and Coursera, among others.
2. Look at 3 false assumptions about big data [slide 4]:
    1. It's complete and accurate
    2. It tells the whole story
    3. Bigger is better
3. What considerations and tradeoffs arise in the computational manipulation of data? [slide 4]
    1. How do you account for missing data?
    2. How do you certify your sources?
    3. How do you decide which data to include and which to exclude?
    4. How much data is enough? (time is money!)
    5. Are your processing algorithms accurate?
4. What is some of the data needed to successfully fly a space mission? (Possible answer: Knowing all about the spacecraft: speed, direction, amount of fuel/oxygen left.) The same problems that applied to early space missions are some of the same problems faced in dealing with big data.
    - You need to decide which factors to include in your calculations, and which to exclude.
    - You need to decide when to make an assumption for missing data or when to estimate.
    - In writing a program for an early space flight there are many unknown factors using a space craft that has never flown before.
    - It's usually impossible to create a perfect algorithm that can take into account every possibility, so how do you allow for errors and changes?
5. What are some of the calculations needed? (Possible answers: how much fuel to release and with which engines.)
    - They had to run many simulations first to see what would happen under various circumstances.
6. See if anybody knows how NetFlix, movie makers, or Amazon use data about their customers to be more successful. [slide 5] http://www.smartdatacollective.com/bernardmarr/312146/big-data-how-netflix-uses-it-drive-business-success (http://www.smartdatacollective.com/bernardmarr/312146/big-data-how-netflix-uses-it-drive-business-success) and http://www.fastcompany.com/3024655/pitch-perfect-and-how-analytics-are-transforming-movie-marketing (http://www.fastcompany.com/3024655/pitch-perfect-and-how-analytics-are-transforming-movie-marketing)

Businesses like Amazon and NetFlix learn the habits of different customers and make recommendations based on their previous choices and others who share similar characteristics (like Google ads).

See if anybody knows the story of Moneyball (based on a true story) of how a baseball team made decisions based on data analysis to become winners, https://en.wikipedia.org/wiki/Moneyball_(film) (https://en.wikipedia.org/wiki/Moneyball_(film)) and how Vivek Ranadive--who knew little about basketball but owned a multi-million dollar computer processing company and knew how to choose and analyze data--coached his then twelve-year-old daughter's (http://www.newyorker.com/reporting/2009/05/11/090511fa_fact_gladwell) National Junior Championship basketball team to the national championship game. He relied upon his sporting knowledge of soccer and cricket paired with his analytic mindset, to create a system of play which allowed his relatively un-athletic team to excel. From the moment that he used intellect and his business experience to coach an inexperienced team to the championship game, the man who once thought basketball was "mindless" was hooked on the sport. http://www.newyorker.com/magazine/2009/05/11/how-david-beats-goliath (http://www.newyorker.com/magazine/2009/05/11/how-david-beats-goliath)

1. How is data analyzed? **Data analysis requires an algorithm, a plan to collect and process data. [slide 6]**
    1. Generate discussion about what data is collected and how it is analyzed. What is a possible algorithm for making a decision about choosing what movies NetFlix might suggest for a customer?
    2. http://www.smartdatacollective.com/bernardmarr/312146/big-data-how-netflix-uses-it-drive-business-success (http://www.smartdatacollective.com/bernardmarr/312146/big-data-how-netflix-uses-it-drive-business-success) Brainstorm: what other data might they collect? (what's currently popular in that age group, demographic, etc.)

2. Choose one of the options and write an outline of an algorithm: choosing a movie to produce or a sports player to hire. [slide 7]
    1. Describe at least two calculations needed
    2. Describe some of the data you'd need to collect.

   Share and discuss.

Activity 2 (5 min): Present homework from previous day after watching TED talks on data. [slide 8]

If time is short, choose only 1 or 2 of the questions from the homework to be presented to the class and collect the rest to grade.

Journal (5 min)

In your writing journal, map out the steps to answer a specific question or find a solution to solve a specific problem using data.

Options for Differentiated Instruction

Extension Activities:

Data analysis activities from NOAA, NASA, and more! - http://climate-expeditions.org/educators/activities.html (http://climate-expeditions.org/educators/activities.html)

Differentiation Instruction:

What is data acquisition? - http://www.ni.com/data-acquisition/what-is/ (http://www.ni.com/data-acquisition/what-is/)

Data analysis and graphs (with Excel sample) - http://www.sciencebuddies.org/science-fair-projects/project_data_analysis.shtml (http://www.sciencebuddies.org/science-fair-projects/project_data_analysis.shtml)

Collecting and analyzing data - http://ctb.ku.edu/en/table-of-contents/evaluate/evaluate-community-interventions/collect-analyze-data/main (http://ctb.ku.edu/en/table-of-contents/evaluate/evaluate-community-interventions/collect-analyze-data/main)

Using Excel for Handling, Graphing, and Analyzing Scientific Data: A Resource for Science and Mathematics Students - http://academic.pgcc.edu/psc/Excel_booklet.pdf (http://academic.pgcc.edu/psc/Excel_booklet.pdf)

Evidence of Learning

Formative Assessment

Journal day 1:

Given this fictitious data:

A blood drive at the local high school reveals that 20% of the students were HIV positive.

- What is your immediate reaction?
- What questions do you have?

Journal day 2: Describe at least 2 ways that we create meaning out of data.

Homework: Feedback from a TED video on big data

Summative Assessment

Students complete the Data Search and Analysis student activity.

Write an outline of an algorithm to make a data-based decision about what movie to produce or what sports team member to hire.

(http://csmatters.org) 4 - 2

0b100 - 0b10

What are Models and Simulations?

Unit 4. Data Acquisition

Revision Date: Jul 21, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

Students will define and identify models and simulations. They will work in groups to propose a simulation that could be used to investigate a hypothesis.

Outcomes

- Students will identify real-world examples of models and simulations.
- Students will understand that models and simulations are used to generate new knowledge, as well as to formulate, refine, and test hypotheses.
- Students will understand that simulations allow hypotheses to be tested without the constraints of the real world.

Overview

1. Getting Started (5 min)
2. Introduction to Content (10 min)
3. Guided Activities (30 min)
    1. Define and Identify Models and Simulations [10 min]
    2. Use Models and Simulations to Answer Questions [20 min]
4. Wrap Up (5 min)

Source

Some of the ideas in this lesson were adapted from the CS10K community site, https://sites.google.com/site/mobilecsp/lesson-plans/realworldmodels (https://sites.google.com/site/mobilecsp/lesson-plans/realworldmodels)

Learning Objectives

CSP Objectives

- 2.3.1 - Use models and simulations to represent phenomena.
    - 2.3.1C - Models often omit unnecessary features of the objects or phenomena that are being modeled.
    - 2.3.1D - Simulations mimic real-world events without the cost or danger of building and testing the phenomena in the real world.
- 2.3.2 - Use models and simulations to formulate, refine and test hypotheses.
    - 2.3.2A - Models and simulations facilitate the formulation and refinement of hypotheses related to the objects or phenomena under consideration.
    - 2.3.2B - Hypotheses are formulated to explain the objects or phenomena being modeled.
    - 2.3.2C - Hypotheses are refined by examining the insights that models and simulations provide into the objects or phenomena.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
    - 7.1.1F - Public data provides widespread access and enables solutions to identified problems.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
    - 7.3.1J - Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.

Math Common Core Practice:

- MP4: Model with mathematics.

Common Core ELA:

- RST 12.7 - Integrate and evaluate multiple sources of information presented in diverse formats and media
- RST 12.8 - Evaluate the hypotheses, data, analysis, and conclusions in a science or technical text

NGSS Practices:

- 2. Developing and using models

Key Concepts

- Models and simulations are used to generate new knowledge, as well as to formulate, refine, and test hypotheses.
- Simulations allow hypotheses to be tested without the constraints of the real world.

Essential Questions

- How can computational models and simulations help generate new understanding and knowledge?

Teacher Resources

Student computer usage for this lesson is: **optional**

These videos supplement the material covered in this lesson:

- Bill Nye and a scaled model of the solar system (4:17) https://www.youtube.com/watch?v=97Ob0xR0Ut8 (https://www.youtube.com/watch?v=97Ob0xR0Ut8)
- Computer Generated Model of a Solar System (2:41) https://www.youtube.com/watch?v=8z5mwAlxBYc (https://www.youtube.com/watch?v=8z5mwAlxBYc)
- Freedom Tower & WTC Buildings in Minecraft (3:01, start video at 1:09) https://www.youtube.com/watch?v=kWfNcjSw_3c (https://www.youtube.com/watch?v=kWfNcjSw_3c) (Could lead to a discussion of the limitations of modeling in Minecraft)
- Tsunami Propagation, NOAA (1:00) https://www.youtube.com/watch?v=BU9wR-rPpEs&index=7&list=UURvjyjVFLdCHZ6EVeOk1a4w (https://www.youtube.com/watch?v=BU9wR-rPpEs&index=7&list=UURvjyjVFLdCHZ6EVeOk1a4w)
- Interactive Simulation of an Epidemic (click on configuration data to change the parameters) http://nrich.maths.org/content/id/4489/epidemic2.swf (http://nrich.maths.org/content/id/4489/epidemic2.swf)

Lesson Plan

Getting Started (5 min)

- Journal: If I flip a coin 10 times, is it possible to predict exactly how many times it will come up heads?
  - Why or why not? Connect prior knowledge of probability to the use of models and simulations.
- Ask: When you flip a coin, what is the probability of heads? (Students will answer 50% or 1 out of 2, etc.)
- Flip a coin 5 times and record the result. Ask: What is the probability the next result will be heads? (Students should answer 50% again.)
- Discuss with the students the difference between theoretical probability and experimental probability.

Introduction to Content (10 min)

Introduce Vocabulary

- **Theoretical probability** is the exact probability of a situation, taking every factor into account. (The chance of heads is 1 result out of 2 possible outcomes: 1/2 = 50%.)
- **Experimental probability** is basically a guess of the theoretical probability of an event using knowledge of the relative frequency of the event occurring (calculating using test or simulation data: 30 heads / 50 flips = 60%).

Most of the time we need to use experimental instead of theoretical probability. Have students think about this as they answer the following questions with a partner:

- What are the chances that it will rain tomorrow?
- How many hits will your favorite baseball player make in his next game?
- If you toss a crumpled sheet of paper into the recycle bin from across the classroom, how likely is it to make it in?

Discussion: As a whole class, discuss how the students determined their answers to each question. (Did some students want to use the computer to access data or actually perform the paper throwing experiment?)

Use the above examples to define models and simulations to the students:

- Will it rain? Using a meteorological model, a meteorologist will run a program with current data for the forecast (simulation).
- How many hits? Use the player's batting average (the mathematical model) to calculate the number of hits in an average game (the simulation).
- How many "baskets"? The paper and basket are the model; running the trials is the simulation.

(Vocabulary from: http://www.systems-thinking.org/modsim/modsim.htm (http://www.systems-thinking.org/modsim/modsim.htm).)

Guided Activities (30 min)

Define and Identify Models and Simulations [10 min]

Examples of models (do not need to show the entire videos for student understanding):

- Watch this video of a human heart simulation: Multi-scale Multi-physics Heart Simulator (https://www.youtube.com/watch?v=2LPboySOSvo) UT-Heart (5:15) (watch up to 2:00; the rest is interesting but not necessary).
- What's an advantage to having so many data points? What about a disadvantage? (A supercomputer is necessary to run the simulation.)
- How can you test a parachute to be used on Mars? https://www.youtube.com/watch?v=_jOzxEOlDJg (https://www.youtube.com/watch?v=_jOzxEOlDJg) (1:11)? Describe the physical test. Before that test, they create models and simulate on the computer - why? (It is very costly to run a test and to create an actual parachute. First be sure an idea passes a simulated test, then build it.)
- Freedom Tower & WTC Buildings in Minecraft (3:01, start video at 1:09) https://www.youtube.com/watch?v=kWfNcjSw_3c (https://www.youtube.com/watch?v=kWfNcjSw_3c) (Could lead to a discussion of the limitations of modeling in Minecraft.)

Have students list models they have seen (and have interacted with) in each of the following (time permitting, have the students find websites to share):

- Entertainment (e.g., a dragon for a movie)
- Military (e.g., a tank or airplane)
- Medical (e.g., bacteria or the human body)

Examples of Simulations:

- Tsunami Propagation, NOAA (1:00) https://www.youtube.com/watch?v=BU9wR-rPpEs&index=7&list=UURvjyjVFLdCHZ6EVeOk1a4w (https://www.youtube.com/watch?v=BU9wR-rPpEs&index=7&list=UURvjyjVFLdCHZ6EVeOk1a4w)
- Interactive Simulation of an epidemic (click on configuration data to change the parameters) http://nrich.maths.org/content/id/4489/epidemic2.swf (http://nrich.maths.org/content/id/4489/epidemic2.swf)

Have students find and share simulations in each of the following:

- Financial (e.g., stock market forecasting)
- Weather (e.g., predicting the path of hurricanes)
- Space (e.g., predicting the path of an asteroid)

Use Models and Simulations to Answer Questions [20 min]

In this activity, the class will match a person and a character in a contest and propose models and simulations to predict the winner.

Have the class suggest people, characters, and activities to fill the chart (start with a blank chart; entries below are for example only):

Person	**Character**	**Activity**
RG III	Sponge Bob	Ping-pong
Bill Gates	Elektra	Jeopardy
Shakira	Harry Potter	Cooking

- Select one item from each column and propose a competition: What if Bill Gates played Harry Potter in a game of ping-pong -- who would win?
- Students should propose an answer (their hypothesis).
- As a class, what characteristics of the person (that can be evaluated and quantified), the character, and the activity helped them choose their answer (such as athleticism, coordination, height for the models, or different serves and returns for the game play)?
- Have students work in pairs to identify the models (characteristics of the person and character) and the simulation (the game play) that would be incorporated into a program to determine the winner. After a few minutes, merge pairs into small groups to compare. As a class, discuss what characteristics do not need to be taken into account for the modeling and simulation. Did anyone worry about hair color, age, or the weather?
- Give each group a large sheet of paper and markers. Have the students select a new person, character, and activity. Create descriptions of the models and simulation in a new program to determine the answer.
- After 10 minutes, each group should post their proposal and have the class complete a "Gallery Walk" to compare and contrast the results.

Wrap Up (5 min)

Journal: Have students record the definitions (in their own words) of the vocabulary used in this lesson: probability, model, simulation, and hypothesis.

Evidence of Learning

Formative Assessment

- Can students define models and simulations in their own words (and understand the difference)?
- During the activity, are students able to identify particular characteristics that will be included in a model and simulation as well as characteristics that are to be excluded?

Internal Use Only

Pending Tasks

CK to do a resource check/review

(http://csmatters.org) 4 - 3

0b100 - 0b11

Using Data and Simulations

Unit 4. Data Acquisition

Revision Date: Jul 20, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Summary

Students will formulate a hypothesis, run simulations, and analyze the results to determine what needs to be modified in their hypothesis and/or the simulation itself.

Outcomes

- Students will identify and create real-world examples of models and simulations.
- Students will use models and simulations to generate new knowledge and to formulate, refine, and test hypotheses.
- Students will use simulations to test hypotheses without the constraints of the real world.

Overview

1. Getting Started (10 min)
2. Guided Activities (35 min)
    1. Activity 1 - Rolling Dice Simulation, materials - large paper and markers or other method to share group work [15 minutes]
    2. Activity 2 - Using a Simulation to Test a Hypothesis [20 minutes]
3. Wrap Up (5 min)

Source

The coin flipping extension is based on a CS10K lesson: https://sites.google.com/site/mobilecsp/lesson-plans/lp-coinflip-miniprojects (https://sites.google.com/site/mobilecsp/lesson-plans/lp-coinflip-miniprojects)

Learning Objectives

CSP Objectives

- 1.2.1 - Create a computational artifact for creative expression.
    - 1.2.1B - Creating computational artifacts requires understanding of and use of software tools and services.
    - 1.2.1C - Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming integrated development environments (IDEs), spreadsheets, three-dimensional (3-D) printers, or text editors.
- 1.2.4 - Collaborate in the creation of computational artifacts.

    - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
- 2.3.1 - Use models and simulations to represent phenomena.
    - 2.3.1A - Models and simulations are simplified representations of more complex objects or phenomena.
    - 2.3.1B - Models may use different abstractions or levels of abstraction depending on the objects or phenomena being posed.
    - 2.3.1C - Models often omit unnecessary features of the objects or phenomena that are being modeled.
    - 2.3.1D - Simulations mimic real-world events without the cost or danger of building and testing the phenomena in the real world.
- 2.3.2 - Use models and simulations to formulate, refine and test hypotheses.
    - 2.3.2A - Models and simulations facilitate the formulation and refinement of hypotheses related to the objects or phenomena under consideration.
    - 2.3.2B - Hypotheses are formulated to explain the objects or phenomena being modeled.
    - 2.3.2C - Hypotheses are refined by examining the insights that models and simulations provide into the objects or phenomena.
    - 2.3.2D - The results of simulations may generate new knowledge and new hypotheses related to the phenomena being modeled.
    - 2.3.2E - Simulations allow hypotheses to be tested without the constraints of the real world.
    - 2.3.2F - Simulations can facilitate extensive and rapid testing of models.
    - 2.3.2G - The time required for simulations is impacted by the level of detail and quality of the models and the software and hardware used for the simulation.
    - 2.3.2H - Rapid and extensive testing allows models to be changed to accurately reflect the objects or phenomena being modeled.
- 3.1.2 - Collaborate when processing information to gain insight and knowledge.
- 4.1.1 - Develop an algorithm for implementation in a program.
    - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
    - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
    - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
    - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
    - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
    - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
    - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
    - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
    - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
    - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
    - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
    - 5.1.1E - A computer program or the results of running a program may be rapidly shared with a large number of users and can have widespread impact on individuals, organizations, and society.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
    - 5.1.2G - Program development includes identifying programmer and user concerns that affect the solution to problems.
    - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.
    - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.3.1 - Use abstraction to manage complexity in programs.
    - 5.3.1A - Procedures are reusable programming abstractions.
    - 5.3.1B - A procedure is a named grouping of programming instructions.
    - 5.3.1C - Procedures reduce the complexity of writing and maintaining programs.
    - 5.3.1D - Procedures have names and may have parameters and return values.
    - 5.3.1H - Data abstraction provides a means of separating behavior from implementation.
    - 5.3.1K - Lists and list operations, such as add, remove, and search, are common in many programs.
    - 5.3.1L - Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
    - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
    - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
    - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.

2.3.1 ABCD

2.3.2 ABCDEFGH

Math Common Core Practice:

- MP2: Reason abstractly and quantitatively.
- MP3: Construct viable arguments and critique the reasoning of others.

Common Core Math:

- F-BF.1-2: Build a function that models a relationship between two quantities
- S-ID.1-4: Summarize, represent, and interpret data on a single count or measurement variable

Common Core ELA:

- RST 12.7 - Integrate and evaluate multiple sources of information presented in diverse formats and media
- RST 12.8 - Evaluate the hypotheses, data, analysis, and conclusions in a science or technical text
- RST 12.9 - Synthesize information from a range of sources

NGSS Practices:

- 3. Planning and carrying out investigations
- 4. Analyzing and interpreting data
- 5. Using mathematics and computational thinking
- 6. Constructing explanations (for science) and designing solutions (engineering)
- 8. Obtaining, evaluation, and communicating information

Key Concepts

Students will be able to:

- identify and create real-world examples of models and simulations.
- use models and simulations to generate new knowledge and to formulate, refine, and test hypotheses.
- use simulations to test hypotheses without the constraints of the real world.

Essential Questions

- How can computational models and simulations help generate new understanding and knowledge?
- How can computation be employed to help people process data and information to gain insight and knowledge?
- How are algorithms implemented and executed on computers and computational devices?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

The PowerPoint "Using Data and Simulations" can be found in the Lesson Resources folder.

Penny Bias article to go with lesson extension: http://mathtourist.blogspot.com/2011/02/penny-bias.html (http://mathtourist.blogspot.com/2011/02/penny-bias.html)

For the Monty Hall Problem extension:

Online simulation of the problem: http://math.ucsd.edu/~crypto/cgi-bin/MontyKnows/monty2?1+17427 (http://math.ucsd.edu/~crypto/cgi-bin/MontyKnows/monty2?1+17427)

There are several videos on YouTube demonstrating and explaining the Monty Hall Problem.

An animated video: https://www.youtube.com/watch?v=mhlc7peGlGg (https://www.youtube.com/watch?v=mhlc7peGlGg) length is 5:48

Live action video: https://www.youtube.com/watch?v=4Lb-6rxZxx0 (https://www.youtube.com/watch?v=4Lb-6rxZxx0) length is 5:30

There is sample code for the die Python program in the Lesson Resources Folder called 4-3 Sample Code.py

Lesson Plan

Getting Started (10 min)

Journal:

a. Describe a situation that can be modeled by rolling one die.
b. Describe a situation that can be modeled by rolling two dice.
c. (Consider possible outcomes and the likelihood of each outcome.)

Have students share their answers with the class.

Guided Activities (35 min)

Activity 1 (15 min) - Designing a Simulation (rolling dice)

1. Arrange the class into small groups (two or three).
2. The groups are to develop an algorithm and write a program to simulate rolling a standard six-sided die.
    - The program should ask for the number of rolls and display the results (number of each possible outcome that was rolled).
    - **Note: The students will need to use the randint function to write this code.**
        - `import random` at the top of the code
        - `random.randint(min,max)` returns a 'random' integer between the min and max values (inclusive).
    - Remind students to use functions where appropriate.
    - Students are encouraged to write their plan (flow chart or pseudocode) on large paper or some other method to enable later sharing with the class.
3. Have a representative from each group describe their plan for the algorithm. (After the first group, subsequent groups could highlight similarities and differences.)
4. Next, have each group discuss and implement changes to their algorithm and programs to simulate rolling two dice. (When providing directions, ask the class how using functions in the previous exercise would make this change easier to implement.)
5. As a class, discuss how many times the program should "roll" the dice. (Be sure students note that every possible value should be displayed, especially upper and lower bounds.)

Activity 2 (20 min) - Using a Simulation to Test a Hypothesis

1. Each group should develop a hypothesis to answer the question: Can a twelve-sided die be used in place of rolling two six-sided dice? Each group should agree on a hypothesis and write it down.
2. The groups already have a design for rolling two six-sided dice. They should now design an algorithm for rolling a twelve-sided die. (This process can involve creating a new function or modifying the current function's parameters to accept the number of sides.)
3. The students should be paired up and begin writing the code necessary to test the hypothesis.
4. Once they finish the code, students should run both simulations to compare results. **Note: Students need to save their code. They will be using it in the next lesson's homework.**
5. As students complete their simulations, if possible, have each pair share their data with the class (in a spreadsheet displayed by the teacher's computer, or on the white board, or large paper).
6. As data is collected, students should determine whether their hypothesis was correct or whether it needs to be modified.
7. Analyze with the class if the simulations themselves were implemented correctly. (Does the data represent the expected theoretical outcomes? How can you modify the program to correct any errors?)

Wrap Up (5 min)

Ask the class: Does your program represent a sufficient simulation for rolling dice? What are the advantages/disadvantages of using a program vs. actual dice?

Journal: Summarize how a program can be used as a simulation to test a hypothesis.

Options for Differentiated Instruction

Students can be provided with the code for a function to simulate rolling one die and use it to develop the rest of the program.

After the first group activity, the teacher can swap a student from each group to allow different input into the next group activity.

Extension #1 - Penny Flipping

This extension is based on advanced mini project # 4, which can be found here:
https://docs.google.com/a/smcps.org/document/d/1AKHpiQ87bE4W1YzHlAFh2uNAHuEtdMOCQVV6HfxfDzc/edit (https://docs.google.com/a/smcps.org/document/d/1AKHpiQ87bE4W1YzHlAFh2uNAHuEtdMOCQVV6HfxfDzc/edit)

Students read an article about the 'randomness' of flipping a penny: http://mathtourist.blogspot.com/2011/02/penny-bias.html (http://mathtourist.blogspot.com/2011/02/penny-bias.html)

Next, students should hypothesize the results of lining up 10 pennies on edge and knocking them over (as described in the article). Students need to determine how many times to run the experiment, collect data, and analyze the results.

Students should work in pairs to write a computer simulation for the penny experiment. (Note: this is a program based on experimental data, not theoretical.)

Discuss as a class the validity of the simulation written. Can this simulation be used for other coins?

Extension - The Monty Hall Problem

In the game show "Let's Make a Deal", the original host was Monty Hall. Onvery show, Monty would present a player with three doors or curtains to choose from. The contestant was asked to choose a door in search of a prize. After making a selection, Monty Hall would open one of the doors not selected by the contestant to reveal a non-prize (perhaps a goat). Then Monty would ask if the contestant wanted to change their choice.

After explaining the show to the class ask, "Should the contestant change?" Students should propose a hypothesis.

Have the students design a simulation to test their hypothesis (discuss what is the data collected and the number of times the simulation should run to collect data). After running the simulation, students should evaluate their hypothesis and determine whether it needs to be modified or whether the simulation needs to be modified.

If Monty Hall had four doors, what should the contestant do?

What should the contestant do if they know that Monty does not know what is behind each door?

Online simulation of the problem: http://www.math.ucsd.edu/~crypto/Monty/monty.html (http://www.math.ucsd.edu/~crypto/Monty/monty.html)

There are several videos on YouTube demonstrating and explaining the Monty Hall Problem.

An animated video: https://www.youtube.com/watch?v=mhlc7peGlGg (https://www.youtube.com/watch?v=mhlc7peGlGg) length is 5:48

Live action video: https://www.youtube.com/watch?v=4Lb-6rxZxx0 (https://www.youtube.com/watch?v=4Lb-6rxZxx0) length is 5:30

Evidence of Learning

Formative Assessment

Review student journal entries and class discussions to determine students' understanding of simulations, a hypothesis, and the ability to determine a method to test a hypothesis.

Summative Assessment

Describe an algorithm to simulate drawing an ace of any suit from a standard deck of cards.

Make a hypothesis about drawing cards from a standard deck of cards and determine how to collect data to answer your hypothesis.

(http://csmatters.org) 4 - 4

0b100 - 0b100

File Input and Output using Python

Unit 4. Data Acquisition

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary

This lesson introduces students to reading information from an input file and writing to an output file as a functionality of Python programming. The students will then apply these concepts to program a simple Dice Roll application to generate data. This lesson will prepare students to read and write files for use in later Data Acquisition lessons.

Outcomes

- Students will learn how to read from an input data file and write to an output file using the Python programming language.
- Students will be able to apply their new knowledge to writing independent programs.

Overview

Session 1

1. Getting Started (10 min) - Journal on the possible advantages/disadvantages of being able to input/output data.
2. Book activity (40 min) - Students are guided through the properties of file I/O using the "Python for Informatics" book.

Session 2

1. Python Lab (50 min) - Students write a program to roll 2 six-sided dice and one 12-sided die and write the output to a file. They then import it to Excel as a txt or csv file and compare distributions using `countif`. (This exercise may be assigned as homework if students have the computing resources to complete a programming assignment as homework.)

Learning Objectives

CSP Objectives

- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
- 2.3.1 - Use models and simulations to represent phenomena.
- 3.2.1 - Extract information from data to discover and explain connections or trends
- 3.3.1 - Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
  - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
  - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
  - 4.1.1E - Algorithms can be combined to make new algorithms.
  - 4.1.1F - Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.
  - 4.1.1H - Different algorithms can be developed to solve the same problem.
  - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
  - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
  - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
  - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
  - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
  - 4.1.2I - Clarity and readability are important considerations when expressing an algorithm in a language.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
  - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
  - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
  - 5.1.1C - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution.
- 5.1.2 - Develop a correct program to solve problems.
  - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
  - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
  - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
  - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.
  - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.3.1 - Use abstraction to manage complexity in programs.
  - 5.3.1A - Procedures are reusable programming abstractions.
  - 5.3.1B - A procedure is a named grouping of programming instructions.
  - 5.3.1C - Procedures reduce the complexity of writing and maintaining programs.
  - 5.3.1D - Procedures have names and may have parameters and return values.
  - 5.3.1G - Parameters provide different values as input to procedures when they are called in a program.
  - 5.3.1H - Data abstraction provides a means of separating behavior from implementation.
  - 5.3.1K - Lists and list operations, such as add, remove, and search, are common in many programs.
  - 5.3.1L - Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.
- 5.4.1 - Evaluate the correctness of a program.
  - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
  - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.
  - 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
  - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
  - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
  - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.

5.1.2 ABCEI

Key Concepts

The students must understand how to open and read from an input file using Python.

The students must understand how to declare and write to an output file using Python.

Essential Questions

- How can computation be employed to help people process data and information to gain insight and knowledge?
- How do people develop and test computer programs?

Teacher Resources

Student computer usage for this lesson is: **required**

Python for Informatics by Charles Severance, http://www.pythonlearn.com/book.php (http://www.pythonlearn.com/book.php).

Explanation of the CountIf function in Excel https://support.office.com/en-us/article/COUNTIF-function-e0de10c6-f885-4e71-abb4-1f464816df34 (https://support.office.com/en-us/article/COUNTIF-function-e0de10c6-f885-4e71-abb4-1f464816df34).

The mbox.txt and mbox-short.txt files are in the Lesson Resources Folder.

Lesson Plan

Session 1

Getting Started (10 min)

Think-Pair-Share

- List some pros and cons of inputting data for a program using only a keyboard?
- List some pros and cons of displaying output of a program using only video display?

Have students review their journal entries as a class and note the advantages and disadvantages on a white board.

Guided Activity (40 min)

Before Starting The Activity:

- Have the students open the book *Python for Informatics* by Charles Severance http://www.pythonlearn.com/book.php (http://www.pythonlearn.com/book.php) and navigate to Chapter 7 "Files" (page 79).
- Students should also open PyCharm or the Runestone coding environment and create a new .py file. (Teachers may want to project PyCharm and the textbook on the board.)
- The two files mbox.txt and mbox-short.txt are located in the Lesson Resources Folder. Students will need to have these saved in the same folder as their python file for the lesson.

Chapter 7: Files

The students should code the examples in the book as the teacher proceeds through the lessons.

1. **Section 7.1 Persistence**
   a. Emphasize the disadvantages of the loss of information when the computer is powered off and the frequent need to store data in a more permanent location.
   b. Introduce the concept of secondary memory.
2. **Section 7.2 Opening Files**
   a. After reading the brief text in Section 7.2, have students type the sample code to open the mbox.txt file and run it.
   b. Have students try to open another nonexistent file and see what kind of error they get.
   c. **Check the student's code for understanding:**
      - Were the students able to open the mbox.txt file successfully?
      - Did the students get a "No such file or directory" error message when they attempted to open a nonexistent file?
3. **Section 7.3 Text Files and Lines**
   a. Introduce the newline special character "end of line" which breaks the file into lines. In Python the newline is represented by "\n" in string constants.
   b. Have the students type in the sample code in 7.3 (page 81), which demonstrates the newline character's function.
   c. **Check the student's code for understanding**
      - Did the "\n" cause the string to be displayed on two lines?
4. **Section 7.4 Reading Files**

a. Introduce the notion of reading each line in a file by using a while loop. (The students should already be familiar with for and while loops, but may need a quick review of the syntax.)
b. Have students count the lines in the mbox.txt file by running the first sample code in 7.4 (page 82).
c. **Check the students' code for understanding.**
   - Does your code give the same output as in the book?

5. **Section 7.8 Writing Files**
   a. Show the students how to open a file for writing by opening it with mode 'w' as a second parameter.
   b. After reading the brief text in Section 7.8, have students type the sample code to write to the output.txt file and run it. Remind them to use the newline character, "\n", at the end of each line.
   c. **Check the student's code for understanding:**
      - Were the students able to write to the output.txt file successfully?
      - Did the students remember to close their output file when finished?

Session 2

Python Lab (50 min)

Preparation:

- Before doing the lab/homework students need to know how to use the `countIf` function in Excel.
- **Example:** `this = COUNTIF(A1:A1000,1)` counts how many 1s are in the range A1 to A1000. You can show the example on the Microsoft office help website. https://support.office.com/en-us/article/COUNTIF-function-e0de10c6-f885-4e71-abb4-1f464816df34 (https://support.office.com/en-us/article/COUNTIF-function-e0de10c6-f885-4e71-abb4-1f464816df34)

Part 1:

- Open the Python program(s) for rolling two six sided and one twelve-sided dice (students should have saved this from lesson 4-3).
- Edit the code so it rolls the dice 1000 times and write the results of each roll to an output file named diceRolls.dat.

Part 2:

- Import the diceRolls.dat file into your spreadsheet software (such as Excel) either as text or cvs file type.
- Make use of the `countif` function to compare the distribution of the rolls for how many times each number 2 through 12 was rolled with the pair of six-sided dice to the distribution for the 12-sided die.
- Show the comparison visually with an appropriate chart in your spreadsheet software.

Options for Differentiated Instruction

Have students work in pairs as the new concepts are introduced and practiced.

For a class needing more scaffolding: Work as a group. Have students take turns around the room to read aloud the brief text in each section in Chapter 7. Do the short exercises together with a "row captain" assigned to each row (or group) in the classroom who is in charge of checking that everybody in their row has completed each short task and has gotten the help needed to finish. Row captains help each other until the entire class has successfully completed each task. Report out on what challenges were encountered, recording problems and solutions at the front of the classroom as the class works. Rotate the role of row captain for each section.

For more independent students: Introduce/demonstrate the key ideas first and then allow student to work through Chapter 7 at their own pace.

Evidence of Learning

Formative Assessment

The teacher will check the student's code for understanding.

The teacher will check for understanding as each new concept is introduced.

Summative Assessment

Exercise 7.1 Write a program to read through a file and print the contents of the file (line by line) all in upper case. Executing the program will look as follows:

python shout.py

Enter a file name: mbox-short.txt

FROM STEPHEN.MARQUARD@UCT.AC.ZA SAT JAN 5 09:14:16 2008

RETURN-PATH: <POSTMASTER@COLLAB.SAKAIPROJECT.ORG>

RECEIVED: FROM MURDER (MAIL.UMICH.EDU [141.211.14.90])

BY FRANKENSTEIN.MAIL.UMICH.EDU (CYRUS V2.3.8) WITH LMTPA;

SAT, 05 JAN 2008 09:14:16 -0500

You can download the sample input file from www.py4inf.com/code/mbox-short.txt (http://www.py4inf.com/code/mbox-short.txt)

Exercise 7.2 Write a program to prompt for a file name, and then read through the file and look for lines of the form:

X-DSPAM-Confidence: 0.8475

When you encounter a line that starts with “X-DSPAM-Confidence:” pull apart the line to extract the floating point number on the line. Count these lines and the compute the total of the spam confidence values from these lines. When you reach the end of the file, print out the average spam confidence.

Enter the file name: mbox.txt

Average spam confidence: 0.894128046745

Enter the file name: mbox-short.txt

Average spam confidence: 0.750718518519

Test your file on the mbox.txt and mbox-short.txt files.

(http://csmatters.org) 4 - 5

0b100 - 0b101

Data Collection, Analysis, and Simulation

Unit 4. Data Acquisition

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Pre-lesson Preparation

Your students will need computers for this lesson. If you would like to show students a working dartboard simulation (http://mw2.concord.org/public/student/game/dartboard.html) (with a circular dartboard), check that your browser can run a Java plug-in. Be sure to update, activate, and disable the plug-in as needed for security purposes.

Summary

In this lesson, students will explore basic data analysis concepts in Python, learn about code extensibility, create a simple simulation from scratch, and reuse their code to make a more elaborate simulation.

Outcomes

- Students will create a mathematical simulation and understand how programming can be used to model real-world processes.
- Students will understand extensibility and code reuse by developing a simulation and modifying it to solve a more complex task.
- Students will be able to reason about and solve a problem by programming a solution from scratch.
- Students will collect and analyze data.

Overview

Session 1:

1. Getting Started (5 min)
2. Guided Activity (45 min)
3. Optional Homework

Session 2:

1. Getting Started (5 min)
2. Guided Activity (45 min)
3. Homework

Part of this lesson was adapted from http://www.nzmaths.co.nz/resource/dartboards (http://www.nzmaths.co.nz/resource/dartboards) and http://www.nzmaths.co.nz/resource/more-dartboards (http://www.nzmaths.co.nz/resource/more-dartboards).

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
  - 1.2.2B - A creative development process for creating computational artifacts can be used to solve problems when traditional or prescribed computing techniques are not effective.
- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
  - 1.2.3A - Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
  - 1.2.3B - Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision.
  - 1.2.3C - Combining or modifying existing artifacts can show personal expression of ideas.
- 1.2.5 - Analyze the correctness, usability, functionality, and suitability of computational artifacts.
  - 1.2.5A - The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact.
  - 1.2.5B - A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact.
  - 1.2.5C - The functionality of a computational artifact may be related to how it is used or perceived.
  - 1.2.5D - The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.
- 2.1.1 - Describe the variety of abstractions used to represent data.
  - 2.1.1A - Digital data is represented by abstractions at different levels.
- 2.3.1 - Use models and simulations to represent phenomena.
  - 2.3.1A - Models and simulations are simplified representations of more complex objects or phenomena.
- 3.1.1 - Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.
  - 3.1.1A - Computers are used in an iterative and interactive way when processing digital information to gain insight and knowledge.
  - 3.1.1B - Digital information can be filtered and cleaned by using computers to process information.
  - 3.1.1D - Insight and knowledge can be obtained from translating and transforming digitally represented information.
  - 3.1.1E - Patterns can emerge when data is transformed using computational tools.
- 3.1.3 - Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.
  - 3.1.3A - Visualization tools and software can communicate information about data.
  - 3.1.3C - Summaries of data analyzed computationally can be effective in communicating insight and knowledge gained from digitally represented information.
  - 3.1.3D - Transforming information can be effective in communicating knowledge gained from data.
- 3.2.1 - Extract information from data to discover and explain connections or trends
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
  - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
  - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
  - 4.1.1E - Algorithms can be combined to make new algorithms.
  - 4.1.1F - Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.
  - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
  - 4.1.1H - Different algorithms can be developed to solve the same problem.
  - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
  - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
  - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
  - 4.1.2F - The language used to express an algorithm can affect characteristics such as clarity or readability but not whether an algorithmic solution exists.
  - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
  - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
  - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.

    - 5.1.1D - Additional desired outcomes may be realized independently of the original purpose of the program.
    - 5.1.1E - A computer program or the results of running a program may be rapidly shared with a large number of users and can have widespread impact on individuals, organizations, and society.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
    - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
    - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
- 5.3.1 - Use abstraction to manage complexity in programs.
    - 5.3.1A - Procedures are reusable programming abstractions.
    - 5.3.1B - A procedure is a named grouping of programming instructions.
    - 5.3.1C - Procedures reduce the complexity of writing and maintaining programs.
    - 5.3.1D - Procedures have names and may have parameters and return values.
    - 5.3.1G - Parameters provide different values as input to procedures when they are called in a program.
    - 5.3.1L - Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1A - Program style can affect the determination of program correctness.
    - 5.4.1B - Duplicated code can make it harder to reason about a program.
    - 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
    - 5.4.1D - Longer code segments are harder to reason about than shorter code segments in a program.
    - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
    - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.
    - 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
    - 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
    - 5.4.1I - Programmers justify and explain a program's correctness.
    - 5.4.1J - Justification can include a written explanation about how a program meets its specifications.
    - 5.4.1K - Correctness of a program depends on correctness of program components, including code segments and procedures.
    - 5.4.1L - An explanation of a program helps people understand the functionality and purpose of it.
    - 5.4.1M - The functionality of a program is often described by how a user interacts with it.
    - 5.4.1N - The functionality of a program is best described at a high level by what the program does, not at the lower level of how the program statements work to accomplish this.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
    - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
    - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
    - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.

2.1.1 A

2.3.1 ABC

3.1.1 ABE

3.1.3 ACD

3.2.1 F

4.1.1 ABCDEFHJ

5.1.1 ACD

5.4.1 BCDFGIJKLMN

Key Concepts

The development of a program from scratch to solve a specific problem is presented to students by creating a simulation that lets them see how software can model a real-world process. Additionally, the concepts of extensibility and code reuse are shown through hands-on programming experience.

Essential Questions

- How are algorithms implemented and executed on computers and computational devices?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?

Teacher Resources

Student computer usage for this lesson is: **required**

The Lesson Resources folder contains an example program showing how to use Python's random function to simulate tossing a coin, as well as a Dartboard.py solution. There is also a comparison between this python code, and some pseudocode that one may write before coding Dartboard.py, titled "Pseudocode vs Python."

An alternative lesson outline using Runestone and PyCharm to code a simulation and use it to develop, refine and test hypotheses in is the lesson folder. The lesson is in a file named "Monte Carlo Simulation to Calculate Pi.docx".

Lesson Plan

Session 1

Getting Started (5 min)

Think-Pair-Share: Writing programs "from scratch"

- Ask your students to think about what it means to write a program "from scratch," how and why someone would do so.
- Have your students pair off to share and list their ideas.
- Collect and discuss their responses. Some topics to note could include: the importance of programming to solve a specific problem, the challenge of starting a program without copying someone else's code, and the value of learning to implement algorithms on one's own. The dartboard simulation used in this lesson provides a good example of a straightforward problem with a solution that someone can code on their own from scratch.

Guided Activity (45 min)

A Dartboard Simulation [10 min]

Get students' attention by asking them to play with the Dartboard Simulator (http://mw2.concord.org/public/student/game/dartboard.html) (requires the Java browser plug-in) as they think about the following scenario and answer the related questions:

- You have a circular dartboard consisting of three concentric circles. The largest outer circle has a radius of 3 units and is colored yellow. Inside that is the middle circle with a 2-unit radius, colored orange. In the center is a red circle, the bull's-eye, with a 1-unit radius. The final board looks like a red circle enclosed by an orange ring, enclosed by a larger yellow ring.
- Now suppose you have a robot that throws darts randomly. How many darts would you expect to hit the bull's-eye if the robot throws 90 darts at your dartboard, assuming every dart hits the board? How many of those darts (on average) would hit the orange part of the board? How many would hit the yellow part?
- To make the dartboard into a game, a certain number of points is awarded each time a dart hits a specific color. You want to assign score points to the dartboard fairly (for our random robot), such that the robot gets a score proportional to the chance it has of hitting a given color on the board. How many points should landing a dart on red be worth? On orange? On yellow?
- Answers:
  - Recall that the formula for the area of a circle is A = pi*radius^2. The red circle has an area of pi units. The orange ring has an area of 3*pi units (4*pi minus the area of the red circle). The yellow ring has an area of 5*pi units (9*pi minus 3*pi for orange and pi for red). The total area is 9*pi. Thus, on average, 50 of the darts would land on yellow, 30 of the darts would land on orange, and 10 darts would land on red.
  - For score assignment, since a single dart would hit yellow 5/9 of the time, orange 3/9 of the time, and red 1/9 of the time, we know that red is 5 times harder to hit than yellow and 3 times harder to hit than orange. Thus, a convenient assignment would be 15 points for red, 5 points for orange, and 3 points for yellow. With these score assignments, if the robot throws a dart it would score 5 points on average.

Devising a Dartboard Simulation [10 min]

Suppose you want to write a program that simulates tossing virtual darts. Each dart will land at a point on a ***square*** virtual dartboard that is one unit long on each side. Each point on this dartboard has both an x and a y coordinate, both of which are between 0 and 1. The bull's-eye is a square in the center of the dartboard with sides of length 0.5 units.

Have the program ask the user how many darts they want thrown. The program should then simulate throwing these darts by generating a random landing location (a random x and a random y coordinate) on the dartboard for each dart. Recall how to use Python's random functions (by reviewing the previous lessons' dice simulation). As darts are thrown, the program counts how many darts land within the center square, the bull's-eye. The bounds of bull's-eye are [0.25, 0.75] on both the x and y axes. Finally, the program should print out the number of darts thrown and the number that landed within that rectangle.

Have your students answer the following questions:

1. Is the problem description clear enough that you could code this program on your own? If so, what information helps to make it clear? If not, what further explanations are needed?
2. What are some variables you think your program will need, and what will you use them for?
3. Are there any built-in Python functions that might be necessary or useful? If so, which ones?

4. What other types of Python constructs are required for this program? Specifically, will you need an input assignment? A loop? An if statement? A print statement? What else?
5. If you do need to use a loop, would it be better to use a while loop or a for loop in this program? Or will either work equally well? Why? If you think you can code the program without using a loop, explain why.
6. Think back to the dice simulation you made. What are two or three ways in which your dartboard program will be similar to the dice simulation? What are two or three ways they will be different?
7. What percentage of darts thrown randomly at the dartboard will hit the bull's-eye?

Programming the Simulation [25 min]

Take the rest of the class time to have your students begin programming their simulation. If they are not able to finish before the session ends, you may want to assign the program as homework, or devote the beginning of the second session to finishing the program. They will need their programs for the work in the next session.

You may want to remind students how to use Python's random function. The following code may be a useful example:

Example random coin flipping code (the python file is available in the Lesson Resources folder):

```
import random # Needed for random number generation

number_of_heads = 0

for i in range(0, 100):

    x = random.random() # Generates a random floating point (decimal) number between 0 and 1

    if x > 0.5:

        number_of_heads = number_of_heads + 1

print "The number of heads in 100 coin flips is ", number_of_heads
```

Optional Homework

Have your students finish their dartboard programs, as they are needed in the next session. Alternatively, if they have finished their programs, you could assign the "Collecting and Analyzing Data" think-pair-share of the next session as a homework, to be discussed in the next session.

Session 2

Getting Started (5 min)

Journal: Making your programs extensible

- Define "extensibility" for your students in the context of a program. Focus on the notion that we should plan for the future by keeping our code simple and organized logically. By doing so, we can more easily extend our code (add new features), reuse parts of our code in other programs, and more swiftly modify our program if the requirements change.
- Have your students answer the following questions in their journals: "What are some reasons to make our programs extensible? What are some things that can go wrong if our code is not extensible?"

Guided Activity (45 min)

Collecting and Analyzing Data [10 min]

Think-pair-share: Collecting and analyzing data

1. Ask your students to recall what percentage of darts randomly thrown will hit the bull's-eye.
2. With the completed square dartboard program, have your students collect data by running their program entering increasing values for the number of darts to be thrown (1, 10, 100, 500, 1000, 5000, 10000, 50000, and so on), recording the output for each input (number of darts that hit the bull's-eye). For each pair of input and output, have them calculate the percentage that hit the bull's-eye (divide output by input, e.g., 12 hits / 50 thrown = 0.24, or 24% of the darts were hits).
3. Have your students pair off and compare their data with that of their partner. Ask them to write down, for any trend they may notice in the data, why they think it is there. Have them recall when they calculated the probability of the robot throwing darts at a circular dartboard.
4. Gather your students and discuss as a class what can be observed from the data. Ideally, as you increase the sample size (number of darts thrown), you should be getting a more accurate measurement of how many darts hit the bull's-eye. As with the dart-throwing robot, the area of the center relative to the whole dartboard determines the percentage that hit. In this case, the center square has an area of 0.25 units^2, and the rest of the dartboard has an area of 0.75 units^2, so approximately 25% of the randomly thrown darts will hit the bull's-eye.

Changing Requirements [30 min]

Discuss with your students how we often want to reuse our code for a new project, and how it is not uncommon when developing a program for the requirements to change. Both of these changes benefit from extensibility in code. Your students will get to test the extensibility of their dartboard program by reusing what they have to fit with a new objective: make a three-ringed circular dartboard.

As before, this program should first ask a user how many darts they would like to throw. Then, it should use that input to simulate throwing darts at a circular dartboard. Finally, it should print the number of darts thrown, the number of darts that hit the bull's-eye, the number that hit the middle ring, the number that hit the outer ring, and the number that missed completely. This circular dartboard is similar to the one from the previous

session's robot exercise: it has a central circular bull's-eye surrounded by a middle ring, which is in turn surrounded by an outer ring. The coordinates for this dartboard are: the center is at coordinate (0,0); the outermost ring is a circle with radius of 3; the middle has a radius of 2; and the bull's-eye has a radius of 1. Simulate throwing a dart by picking random x and y coordinates, each between -3 and 3. Since this range is a square, some darts may miss the dartboard completely. For this program, students may reuse as much of their square dartboard code as they need, but make sure to preserve their original program separately.

Homework

Have your students finish their circular dartboard programs and answer the following questions:

1. How extensible was your square dartboard simulation? How much code were you able to reuse for your circular dartboard simulation, and what did you need to write from scratch?
2. What are the major differences between the code in the programs and the way you approached coding them? Which did you find more difficult to program? Would it have been easier or faster to code the circular dartboard program from scratch?
3. Collect data by running the circular dartboard program with various increasing inputs (1, 10, 100, 500, 1000, 5000, and so on) and record the output. For each input and corresponding output, calculate the percentage of darts that missed, that hit the outer ring, that hit the inner ring, and that hit the bull's-eye. What trends do you see in the data? How is the behavior similar and different to the square dartboard program?

Options for Differentiated Instruction

Note: graphics.py may be used with this lesson to create a visualization - http://mcsp.wartburg.edu/zelle/python/ppics2/code/graphics.py (http://mcsp.wartburg.edu/zelle/python/ppics2/code/graphics.py)

Another Note: In order to run graphics.py, you may possibly need to save files in the same Python folder as the program being written due to student access and student space restrictions in your network, (i.e. if the link cannot point back to a network drive file due to restrictions).

Evidence of Learning

Formative Assessment

The students will produce two simulation programs on their own: the square dartboard simulation and the circular dartboard simulation.

Summative Assessment

The students will record their understanding of extensibility in their journal.

The students will gain experience collecting data in both sessions.

The students will think analytically about their programs by answering the questions in Session 1 and in the homework for Session 2.

(http://csmatters.org) 4 - 6

0b100 - 0b110

Optional

Hypothesis Testing with Simulations in NetLogo

Unit 4. Data Acquisition

Revision Date: Jul 18, 2016 (Version 2.0)
Duration: 3 50-minute sessions

Lesson Summary

Note: This optional lesson is designed for advanced students and teachers who would like to introduce another programming tool and environment: NetLogo. Teachers may also choose to complete only the first session (on the basics of NetLogo), to expose students to a new computational platform and way of thinking, and to extend the ideas in Unit 4 about modeling and simulation.

Summary

This lesson teaches students to use simulations to develop and refine hypotheses, then use the simulations to test these hypotheses. NetLogo, which is used throughout the lesson to illustrate the use of functional and data abstraction, is a programmable modeling environment for simulating natural and social phenomena.

NetLogo uses an extension of Logo instead of Python, so students are not expected to write new code in this lesson. See http://www.ianbicking.org/docs/PyLogo_lightning.html (http://www.ianbicking.org/docs/PyLogo_lightning.html) for a comparison of Logo and Python.

Outcomes

- Students will understand that models are abstraction of real environments and will recognize the rationale for and limitations of modeling techniques to analyze problems.
- Students will recognize the use of functional and data abstractions in modeling.
- Students will be able to develop and test hypotheses using an experimental approach in a modeling framework.

Overview

Session 1 - Modeling in NetLogo

- Getting Started (8 min)
- Learning NetLogo (40 min)
- Wrap Up (2 min)

Session 2 - Models and Hypothesis Design

- Getting Started (5 min)
- Models and Hypotheses (20 min)
- Model Selection and Hypothesis Development (25 min)

Session 3 - Hypothesis Testing

- Getting Started (5 min)
- Hypothesis Testing (40 min)
- Wrap Up (5 min)

Learning Objectives

CSP Objectives

- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
  - 1.2.3A - Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
  - 1.2.3C - Combining or modifying existing artifacts can show personal expression of ideas.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4F - A collaboratively created computational artifact can reflect personal expressions of ideas.
- 1.3.1 - Use computing tools and techniques for creative expression.
  - 1.3.1E - Computing enables creative exploration of both real and virtual phenomena.
- 2.2.3 - Identify multiple levels of abstractions used when writing programs.
- 2.3.1 - Use models and simulations to represent phenomena.
- 2.3.2 - Use models and simulations to formulate, refine and test hypotheses.
- 3.1.1 - Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.
- 3.1.2 - Collaborate when processing information to gain insight and knowledge.
- 3.2.1 - Extract information from data to discover and explain connections or trends
- 3.3.1 - Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information.
- 4.1.2 - Express an algorithm in a language.
  - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
  - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
  - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
  - 4.1.2D - Different languages are better suited for expressing different algorithms.
  - 4.1.2E - Some programming languages are designed for specific domains and are better for expressing algorithms in those domains.

    - 4.1.2F - The language used to express an algorithm can affect characteristics such as clarity or readability but not whether an algorithmic solution exists.
    - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
    - 4.1.2I - Clarity and readability are important considerations when expressing an algorithm in a language.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
    - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
    - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
    - 5.1.1C - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution.
    - 5.1.1D - Additional desired outcomes may be realized independently of the original purpose of the program.
    - 5.1.1F - Advances in computing have generated and increased creativity in other fields.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2E - Documentation about program components, such as code segments and procedures, helps in developing and maintaining programs.
    - 5.1.2F - Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments.
- 5.1.3 - Collaborate to develop a program.
    - 5.1.3A - Collaboration can decrease the size and complexity of tasks required of individual programmers.
    - 5.1.3B - Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.
    - 5.1.3F - Effective communication between participants is required for successful collaboration when developing programs.
- 5.2.1 - Explain how programs implement algorithms.
    - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
    - 5.2.1B - Program instructions are executed sequentially.
    - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.

Key Concepts

Students will understand that models are abstraction of real environments and will recognize the rationale for and limitations of modeling techniques to analyze problems.

Students will recognize the use of functional and data abstractions in modeling.

Students will be able to develop and test hypotheses using an experimental approach in a modeling framework.

Essential Questions

- How can computing extend traditional forms of human expression and experience?
- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How can computational models and simulations help generate new understanding and knowledge?
- How are algorithms implemented and executed on computers and computational devices?
- Why are some languages better than others when used to implement algorithms?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

NetLogo (http://ccl.northwestern.edu/netlogo/). http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

Modeling and Simulation 101 video (https://www.youtube.com/watch?v=X-6zxImekOE (https://www.youtube.com/watch?v=X-6zxImekOE))

Lesson Plan

Session One

Getting Started (8 min)

Introduce modeling and simulation using the first four minutes of the Modeling and Simulation 101 video (https://www.youtube.com/watch?v=X-6zxImekOE) (https://www.youtube.com/watch?v=X-6zxImekOE (https://www.youtube.com/watch?v=X-6zxImekOE)). Students open a document for notes for today's session.

Students should record and briefly discuss these four statements about modeling and simulation:

1. Models describe real-world activities using abstration, enabling testing of hypotheses at a fraction of the cost of actual experiments.
2. Models can be expressed in computational, mathematical, textual, and/or graphical forms.
3. Simulations are forms or instances of models that can be implemented as computer programs.
4. Monte Carlo simulations exploit randomness to arrive at their results.

Learning NetLogo (40 min)

To start, all students should download netlogo from this link http://ccl.northwestern.edu/netlogo/ (http://ccl.northwestern.edu/netlogo/).

Students should work through the NetLogo tutorial packet either in groups or as a class. There will be some "thought questions" throughout that students should discuss in their groups and as a class.

Wrap Up (2 min)

Students should complete an exit ticket listing one interesting idea they learned, or one question they have about NetLogo or modeling.

Session Two

Getting Started (5 min)

Review yesterday's NetLogo lesson and ask the students to share what they learned, how NetLogo is similar to or different from Python, and any questions they have about how it works.

Models and Hypotheses (20 min)

Introduction (10 min)

a. Students start NetLogo and open the Art>Fireworks model.
b. Students use the interface Buttons for Setup and Go to run the simulation.
c. Have students read the information in the Info tab, and look through the code in the Code tab, to get a sense of what is being simulated and how it works.
d. Explain that the **model** is the *description* of the **environment,** while the **simulation** is the *specific implementation* of the **model**.
e. **Think -Pair- Share:** Ask the students to identify which aspects of the real-world environment (actual fireworks) ***are*** implemented in the model, and which aspects ***are* not** implemented.
f. Discuss with the class the fact that the simpler characteristics of the model or abstraction make the implementation of the model much easier, at the expense of a possible failure to represent key relationships.
g. Have the students examine the implementation (Code tab) and find the names of the five functions defined in the code. Show where all 5 are called (three in the code and two in the Interface).
h. Discuss these facts about forming a hypothesis
    - A hypothesis is an educated guess about how things work.
    - A hypothesis is written like this: "If _____[I do] _____, then _____[this will result]_____
i. Explore hypotheses that could be tested by this model. (Some hypotheses are suggested by the discussion in the Info tab -- e.g., "If gravity is set to 0, then _____."
j. Ask:
    - What are two things that we can change through the interface?
    - What do we think will happen we we make those changes?

Activity (10 min)

Ask each student to write a hypothesis that can be tested with this simulation, share the hypothesis with elbow partners, and briefly experiment with the parameters to informally test the hypothesis.

Model Selection and Hypothesis Development (25 min)

Note: The "Hypothesis Testing Worksheet" which will be used for the next two lessons is available in the Lesson Resources Folder

For the rest of today's session and Session 3, students will work in teams of four students to select a model to experiment with, then divide into two partner sets to develop a hypothesis, devise an experimental plan, test the hypothesis, and write about their results.

Directions

1. First divide the class into teams of four students, and have them spend 10 minutes exploring different models in NetLogo (which they should already have some familiarity with from Session 1 and the earlier part of Session 2). They should choose one model to focus on and write the name of the model in the corresponding part of the Hypothesis Testing worksheet, along with a short explanation of why they chose the model.
2. For the next 10 minutes, they should study the parameters of the model and how it works, trying things out in the interface to see how changing the parameters affects behavior.
3. In the final 10 minutes of the class session, split the teams into partner groups and have each partner group begin to develop hypotheses about the team's selected model. You may wish to assign them to write these hypotheses down as a homework assignment, either together or individually, using the Hypothesis Testing worksheet to record their hypothesis, why they selected that hypothesis, and the experimental parameters they will use to test the hypothesis.

Session Three

Warm Up (5 min)

Partners should revisit their hypotheses, and choose one hypothesis to focus on first. (They can test both hypotheses if they have time.) Each partner pair should write the name of their model and their selected hypothesis on the board, to share with the other students.

Experimental Design (10 min)

1. Students should use the Hypothesis Testing worksheet to:
    a. Identify an experiment that will test their hypothesis.
    b. Select one or more parameters to vary.
    c. Choose what settings they will use for these parameters.
    d. Determine what measurements they will record.
    e. Decide how many trials they will run at each setting.
    f. **Note:** Be sure that students have completed their experimental design ***before*** proceeding to perform the experiment -- many students will want to just jump in and start trying things, but one goal of the lesson is to help them understand that having a clear hypothesis and experimental design before collecting data is an important part of the scientific process.

Hypothesis Testing (30 min)

For the next twenty to thirty minutes, students should carry out their experiments and record the appropriate measurements.

At the end of the section or for homework, students should write up their findings in a short report, showing the data they've collected (optionally in a graphical form, particularly if assigned as homework), discussing what the data says about their hypothesis, and concluding whether the hypothesis is supported or refuted by the simulation.

Wrap Up (5 min)

Students should come back into their teams to share their findings, and discuss the advantages and disadvantages of using models and simulations to develop and test hypotheses.

Evidence of Learning

Formative Assessment

Students will share and post their hypothesis before testing and sharing the results. Teachers will verify that the hypothesis are falsifiable and testable by the simulations.

Summative Assessment

Students will select a model, develop a hypothesis, design an experiment, and use a simulation to test the hypothesis.

Internal Use Only

Pending Tasks

Caroline to check formatting, design Hypothesis Testing worksheet for lesson, and update/refer to Boids in Motion tutorial handout for use in Session 1.

(http://csmatters.org) 4 - 7

0b100 - 0b111

Unit 4 Assessment

Unit 4. Data Acquisition

Revision Date: Jul 18, 2016 (Version 2.0)
Duration: 1 50-minute sessions

Lesson Summary

Pre-Lesson Preparation: Students need to have already chosen a topic and had it approved by the instructor. Students can use the following sources to help choose a data set:

http://www.data.gov/ (http://www.data.gov/) , http://data.princeton.edu/wws509/datasets (http://data.princeton.edu/wws509/datasets) , http://www.statsci.org/datasets.html (http://www.statsci.org/datasets.html)

Summary:

This lesson is the summative assessment for Unit 4 on Data Analysis. Students will select a data set and write a small Python program to analyze the data. Students will then write a summary of their findings to demonstrate understanding of the data analysis process.

Outcomes:

- This unit assessment is designed to provide more practice in project based-work to prepare students for the final performance project at the end of this class.

Overview:

1. Getting Started (5 min) - Overview of task for the day.
2. Independent Activity (40 min) - Individually or in pairs, students collect and analyze data on their topic.
3. Wrap Up (5 min) - Overview of homework goals and expectations.
4. Homework: Individual two-page summary about their findings.

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
  - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
  - 1.2.3A - Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
- 3.1.1 - Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.
- 3.1.2 - Collaborate when processing information to gain insight and knowledge.
- 3.2.2 - Determine how large data sets impact the use of computational processes to discover information and knowledge.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
  - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
  - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.

- 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
- 4.1.1H - Different algorithms can be developed to solve the same problem.

- 4.1.2 - Express an algorithm in a language.
  - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
  - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
  - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
  - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
  - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
  - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
  - 5.1.1C - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution.
  - 5.1.1D - Additional desired outcomes may be realized independently of the original purpose of the program.
  - 5.1.1E - A computer program or the results of running a program may be rapidly shared with a large number of users and can have widespread impact on individuals, organizations, and society.
- 5.1.2 - Develop a correct program to solve problems.
  - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
  - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
  - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
  - 5.1.2D - Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.
  - 5.1.2E - Documentation about program components, such as code segments and procedures, helps in developing and maintaining programs.
  - 5.1.2G - Program development includes identifying programmer and user concerns that affect the solution to problems.
  - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.
  - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.2.1 - Explain how programs implement algorithms.
  - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
  - 5.2.1B - Program instructions are executed sequentially.
  - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
  - 5.2.1D - An understanding of instruction processing and program execution is useful for programming.
  - 5.2.1G - A process may execute by itself or with other processes.
  - 5.2.1I - Executable programs increase the scale of problems that can be addressed.
  - 5.2.1J - Simple algorithms can solve a large set of problems when automated.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
  - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
  - 5.5.1B - Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.
  - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
  - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
  - 5.5.1F - Compound expressions using and, or, and not are part of most programming languages.
  - 5.5.1J - Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection.

Essentail Knowledge

3.1.1 CDE

3.2.2 BCG

5.1.1 A

5.1.2 ABJ

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP3: Construct viable arguments and critique the reasoning of others.
- MP4: Model with mathematics.
- MP5: Use appropriate tools strategically.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.

Common Core Math:

- S-ID.1-4: Summarize, represent, and interpret data on a single count or measurement variable

- S-ID.5-6: Summarize, represent, and interpret data on two categorical and quantitative variables
- S-ID.7-9: Interpret linear models
- S-IC.3-6: Make inferences and justify conclusions from sample surveys, experiments and observational studies

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- WHST 12.5 - Develop and strengthen writing as needed by planning, revising, editing, rewriting
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products
- WHST 12.7 - Conduct short as well as more sustained research projects to answer a question

NGSS Practices:

- 1. Asking questions (for science) and defining problems (for engineering)
- 2. Developing and using models
- 3. Planning and carrying out investigations
- 4. Analyzing and interpreting data
- 5. Using mathematics and computational thinking

Key Concepts

Students will demonstrate their understanding of the process of collecting and evaluating data.

Essential Questions

- How can computation be employed to help people process data and information to gain insight and knowledge?
- How can computation be employed to facilitate exploration and discovery when working with data?
- What opportunities do large data sets provide for solving problems and creating knowledge?
- How are algorithms implemented and executed on computers and computational devices?
- How are algorithms evaluated?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

Rubric provided on Google Drive - Rubric - Unit 4 Summative Assessment.htm in the lesson folder.

Lesson Plan

Getting Started (5 min)

Verify that every student has selected a topic (approved by the instructor in advance) and address what the goal is for today.

Independent Activity (40 min)

Students will either individually or in pairs (instructor's decision) create a small program that reads data from a file, analyzes it, creates a simple simulation and finally writes data to a file.

Wrap Up (5 min)

Presentation about the expectations of the homework assignments.

Homework

Each student should create a 2-page typed summary that explains the following areas:

- The chosen data
- Why the data topic was chosen

- The analysis process
- The results of the analysis process
- The coding process used for data analysis

Options for Differentiated Instruction

Instructor has the option to have students work individually or in pairs for this assessment.

Evidence of Learning

Formative Assessment

Review Rubric with class and clarify expectations.

Summative Assessment

Students will be assigned a unit project, with a topic of their choice, to demonstrate their understanding and mastery of the concepts of data collection and analysis.

Internal Use Only

Future Work

It would be advisable to also have an objective summative assessment on the concepts presented in this Unit in the future.

(http://csmatters.org) 5 - 1

0b101 - 0b1

Manipulating Large Data Sets

Unit 5. Data Manipulation

Revision Date: Jul 22, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary

Big Data has been defined in many different ways. Easy access to large sets of data and the ability to analyze large data sets changes how people make decisions. Students will explore how Big Data can be used to solve real-world problems in their community. After watching a video that explains how Big Data is different from how we have analyzed and used data in the past, students will explore Big Data techniques in online simulations. Students will identify appropriate data source(s) and formulate solvable questions.

Outcomes

- Students will explain how analyzing Big Data is different from the way ordinary data is analyzed.

- Students will describe how computers can make predictions and answer questions through the use of Big Data, storage of data, and processing data.
- Students will synthesize the relationship(s) between causation and correlation.

Overview

Session 1- What is Big Data?

1. Getting Started (10 min) - Journal
2. Guided Activities (30 min) – Reading and Video
3. Wrap Up (10 min) – Group Review
4. Homework

Session 2 – Where can big data be used?

1. Getting Started (5 min) - Journal
2. Guided Activities (15 min) – Processing Big Data
3. Independent Activities (25 min) – Online Research
4. Wrap Up (5 min) – Exit Slip

Learning Objectives

CSP Objectives

- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
  - 1.2.3A - Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
  - 1.2.3B - Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision.
- 3.1.1 - Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.
  - 3.1.1C - Combining data sources, clustering data, and data classification are part of the process of using computers to process information.
- 3.1.3 - Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.
  - 3.1.3A - Visualization tools and software can communicate information about data.
  - 3.1.3B - Tables, diagrams, and textual displays can be used in communicating insight and knowledge gained from data.
- 3.2.1 - Extract information from data to discover and explain connections or trends
  - 3.2.1A - Large data sets provide opportunities and challenges for extracting information and knowledge.
  - 3.2.1B - Large data sets provide opportunities for identifying trends, making connections in data, and solving problems.
  - 3.2.1C - Computing tools facilitate the discovery of connections in information within large data sets.
  - 3.2.1D - Search tools are essential for efficiently finding information.
  - 3.2.1E - Information filtering systems are important tools for finding information and recognizing patterns in the information.
  - 3.2.1F - Software tools, including spreadsheets and databases, help to efficiently organize and find trends in information.
- 3.2.2 - Determine how large data sets impact the use of computational processes to discover information and knowledge.
  - 3.2.2B - The storing, processing, and curating of large data sets is challenging.
  - 3.2.2C - Structuring large data sets for analysis can be challenging.
  - 3.2.2D - Maintaining privacy of large data sets containing personal information can be challenging.
  - 3.2.2G - The effective use of large data sets requires computational solutions.

3.1.1 C

3.1.3 AB

3.2.1 ABCDEF

3.2.2 BCDG

Essential Questions

- How can computing extend traditional forms of human expression and experience?
- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How can computation be employed to help people process data and information to gain insight and knowledge?
- How can computation be employed to facilitate exploration and discovery when working with data?
- What opportunities do large data sets provide for solving problems and creating knowledge?

Teacher Resources

Student computer usage for this lesson is: **required**

Reading assignment and video clips for Session 1:

- Mind-guessing game: http://en.akinator.com/ (http://en.akinator.com/)
- Random number generator (for choosing students to present): http://www.random.org/ (http://www.random.org/)
- Reading assignment: http://www.foreignaffairs.com/articles/139104/kenneth-neil-cukier-and-viktor-mayer-schoenberger/the-rise-of-big-data (http://www.foreignaffairs.com/articles/139104/kenneth-neil-cukier-and-viktor-mayer-schoenberger/the-rise-of-big-data)
- Moneyball Statistics clip: https://www.youtube.com/watch?v=rMObWsKaIls (https://www.youtube.com/watch?v=rMObWsKaIls) (2:47 mins)
- 3 V's clip: https://www.youtube.com/watch?v=7D1CQ_LOizA (https://www.youtube.com/watch?v=7D1CQ_LOizA)

Possibly useful resource(s) for data collection:

- http://www.gapminder.org/ (http://www.gapminder.org/)
- http://r-dir.com/reference/datasets.html (http://r-dir.com/reference/datasets.html)
- https://dreamtolearn.com/doc/2HDNJH3XJU6CVGKZ7SDM4MCSW (https://dreamtolearn.com/doc/2HDNJH3XJU6CVGKZ7SDM4MCSW)
- https://www.kaggle.com (https://www.kaggle.com) (requires an account - but is free)
- http://catalog.data.gov/dataset (http://catalog.data.gov/dataset)

Big Data Concepts:

- https://youtube.com/watch?v=RG8R1iXEyPc (https://www.youtube.com/watch?v=RG8R1iXEyPc) (16:04 mins) from CS-P Alabama is good background for a teacher to learn more about big data.

Sample data sets (both acquired from http://catalog.data.gov/dataset (http://catalog.data.gov/dataset)) :

- FailedBanklist.csv
- Consumer_Complaints.csv

Lesson Plan

Session 1 - What is Big Data?

Getting Started (10 min) - Journal

Journal: How can a computer gather data from people ? (Think-Pair-Share)

Remind students of the mind guessing game: http://en.akinator.com/ (http://en.akinator.com/) or 20 questions http://www.20q.net/ (http://www.20q.net/)

Discuss: How can the computer learn from people when playing one of these games? How many different answers do you think it could possibly know?

Teacher note: students are not expected to actually play this game during class.

- Students should document in their journal the answer to this question: How does this game store all of the possible answers?
- Ask 3 students to share their answers. (Possible strategies to select a random student: random.com, or pick a random student name stick from a cup.)
- This activity should lead into today's lesson on how large amounts of data are stored and then accessed as needed in a system.

Guided Activities (30 min) - Reading & Video

- Read The Rise of Big Data in chunks: An Introduction to "Big Data" (20 mins)
  - Reading can be found at: http://www.foreignaffairs.com/articles/139104/kenneth-neil-cukier-and-viktor-mayer-schoenberger/the-rise-of-big-data (http://www.foreignaffairs.com/articles/139104/kenneth-neil-cukier-and-viktor-mayer-schoenberger/the-rise-of-big-data)
  - Note: The article is long - break students into groups to read 1-3 paragraph sections, then write the essence of their section as a tweet, as close to 140 characters as possible. Share tweets with the group; require each person in each group share a tweet.
- Point out how big data is everywhere and can be used in many facets of daily life. By using *more* data, we can use processing power to do an analysis that is better and more accurate than using the traditional method of collecting small sets of data and then making assumptions from that data.
- Show/Discuss video clip from the movie Moneyball: (5 mins) https://www.youtube.com/watch?v=rMObWsKaIls (https://www.youtube.com/watch?v=rMObWsKaIls)(movie clip is 2:47 mins)
- As students watch, they are to journal: How is the baseball player data used? Why is the data retrieved?
- Use a random generator (random.org, select by name calling sticks, etc.) to ask 3 students to share their answers.
- Show the first 3-5 minutes of this clip. (It becomes a bit dry, so just show the amount that is appropriate for your students to get the idea): https://www.youtube.com/watch?v=7D1CQ_LOizA (https://www.youtube.com/watch?v=7D1CQ_LOizA) (8:32 total)

1. What are the 3 V's? List some details about each V.
2. (at 4:40) What is Hadoop and how is it used?
3. Identify appropriate data source and form questions
4. Extract data source into format supported by underlying tools
5. Normalize data (remove redundancies, irrelevant details)
6. Import data into tool
7. Perform analysis
8. Visualize results

- Some other concepts to point out to students if there is time:
    - Some examples of how big data is used:
        - Netflix and Amazon use it to improve user recommendations
        - Dominos used it to determine that more people order pizza when it is raining so they now base some of their ad campaigns around weather patterns
        - Help police predict when and where crimes will appear
    - Some examples of how big data was inappropriately used:
        - In 2012 Target store's “outing” a teenager’s pregnancy
        - In 2012 Google spent 22.5 million on a settlement over allegations that they secretly tracked user’s web surfing
        - In 2012 Facebook spent 20 million to settle a lawsuit that alleged that they used user pictures without the user’s knowledge to endorse products that they “liked”
        - In 2013 the revelation of the NSA using big data for national security concerns

Wrap Up (10 min) – Group Review

- Place students in groups of 4 where the first student is A, the next is B, etc. Each group creates a single sheet of paper with the letters across the bottom and the numbers 1 - 5 to demonstrate the level of understanding of each concept. Students should plot their understanding for each respective concept (see list below) on the graph. (The file "BigDataSampleDotGraph" (https://drive.google.com/open?id=1mVPZpte0c9cbwMVODlPd5r3dMu7aLhwdfutKgmxvT30) in the lesson resources folder shows an example.)
- Collect this graph as a level of student’s understanding of the concepts in the video.
- Review each concept using the notes below.
    - A. The three “Vs”: Volume, Variety, and Velocity
        - Big data is kind of like drinking water from a fire hose. It's too much to process for a small pipeline…
    - B. Big Data processing steps:
    - C. Tools for processing big data:
        - Microsoft Excel (or some type of spreadsheet tool - i.e. Calc is another one)
        - Hadoop - a well known big data tool, provided by Apache, requires extensive programming knowledge to set up and use
        - SAS - provides a more intuitive interface and better graphical representations of data
        - Google Prediction API - takes advantage of machine learning to extract meaning from data
        - BitDeli - lightweight, easier to use version of Hadoop
    - D. Very few restrictions on use of big data
        - Companies collect *large* amounts of data on their customers
        - Can be sold to other companies
        - Can be sold to the government
        - Can be used to “de-anonymize” someone

Homework

Students are to pick three topics they want to research that use big data. It is preferred that these topics relate to something learned this year in the course (e.g., the need for IPv6). Tomorrow, as the students enter class, they will sign up on a list with their chosen topic. Since the students will have three options, it is likely they will get one of their selected topics to research.

Session 2 - Where can Big Data be used?

Getting Started (5 min) - Journal

Journal: Think about you daily and weekly activities. What types of data are being stored about you?

Remind students to think about what they do online, in stores, while in a car, etc.

Guided Activities (10 min) – Processing Big Data

Review the steps to processing Big Data:

1. Identify appropriate data source and form questions
2. Extract data source into format supported by underlying tools
3. Normalize data (remove redundancies, irrelevant details)
4. Import data into tool
5. Perform analysis
6. Visualize results

As a class, walk through these steps using the two files in the lesson resources folder (https://drive.google.com/open?id=0By2KZS8SzSUcMkJxbTIzMm9RRm8)(FailedBanklist.csv (https://drive.google.com/open?id=0B2umpE0UajfYaFB6bVlGTmhXYzQ) & Consumer_Complaints.csv (https://drive.google.com/open?id=0B2umpE0UajfYQ05rcEg5MlJhbUE))

Step 1.

Demonstrate how files such as these can be obtained at http://catalog.data.gov/dataset (http://catalog.data.gov/dataset)

Formulate questions such as:
Are there any banks that are on both the complaint list and the failed banklist?
Can we make some deductions about banks that may be on both lists? If so, what deductions can we make?

Step 2.

Extract data source into format supported by underlying tools

Open one of these files in Notepad (or some simple editing program such as Notepad++) and demonstrate how the actual data itself is separated by commas, thus the file name “csv” for comma separated value.

Open both files in Microsoft Excel. Complete a find for the bank name “Banco Popular de Puerto Rico” on both lists. You may want to first sort the data by bank name to find this bank or you can use CTRL + F to find the bank name (see screenshots below).

Step 3.

Normalize data (remove redundancies, irrelevant details)

In this step, there is technically no need to remove redundancies or irrelevant details but you can show the students how you could remove data or limit the data to a particular data set. For example, if were to want to look at only the banks from Maryland, you can use the filter tool to only view those banks from MD.

Step 4.

Import data into tool

Right now the file type is as a csv file. By resaving the file as a .xlsx file it becomes a true spreadsheet file.

Step 5.

Perform analysis

We have determined that the bank “Banco Popular de Puerto Rico” is on both lists. Now ask the students “Why is this bank on both lists?” Note: On the Failed Bank list the Banco Popular de Puerto Rico is actually an acquiring institution. By looking more closely at the dates of the acquisition of the failed bank “Westernbank Puerto Rico” one can formulate some possible deductions that maybe the reason “Banco Popular de Puerto Rico” is on the complaint list is because they had recently taken over a failed bank. It could be possible that some of these complaints were related to this recent acquisition.

Step 6.

Visualize Results

Explain to students that they will learn more about visualize their results in Unit 6. They can complete graph visualization in excel. Show them the website: http://www.gapminder.org/ (http://www.gapminder.org/). Explain that even though a visualization in excel is not interactive like http://www.gapminder.org/ (http://www.gapminder.org/), they can complete some form of visualizing their data by using a spreadsheet. Note: http://www.gapminder.org/ (http://www.gapminder.org/) is VERY attention grabbing. Only briefly show the students what they can do with it (see how data changes over time, look at many different data sets, and download data in different forms - including csv and xlsx formats).

Independent Activity (30 min) – Online Research

Students should research their selected topics from homework. Some possible websites for finding data are listed above under “Possible good resource(s) for data collection.”

Students are to get your approval for a topic and then use the Big Data Sets Worksheet (https://drive.google.com/open?id=1q83owjEkA_RTcmqp5DwaR0zzUEHwrilxiNvzgTbjBAI) in the Lesson Resource Folder (https://drive.google.com/open?id=0By2KZS8SzSUcMkJxbTIzMm9RRm8) to find big data sets that are related to the approved topic.

Wrap Up (5 min) – Exit Slip

Students are to review using http://www.gapminder.org/ (http://www.gapminder.org/) looking specifically at life expectancy. Students will write *one* question after “playing” the timeline of life expectancy using gapminder on an exit slip before leaving class. For example, one may write “Why is the life expectancy of countries such as Denmark, Sweden, & Norway typically higher than other countries throughout most of the timeline?”

Evidence of Learning

Formative Assessment

Students are to submit a document stating their topic for research using Big Data. This document should answer the questions:

Topic:
How is Big Data used to solve or remedy the topic?
Link(s) used to find Big Data? (i.e. data.gov, etc)

Summative Assessment

How has the transformation of data storage affected how data itself is used?

Answer: Storage and processing of large digital data enables us to analyze large data sets quickly rather than small sampling sizes as used before.

How can a computer use Big Data to make predictions?

Answer: Computers can use smart algorithms, powerful processors, and clever software to make inferences and predictions for solvable questions.

(http://csmatters.org) 5 - 2

0b101 - 0b10

Searching

Unit 5. Data Manipulation

Revision Date: Jul 18, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Pre-Lesson Preparation

- Teachers will need to have a piece of paper with a unique number on it for all but one student in the class.
- Students will need access to the datasets and Python skeleton code in the Lesson Resources folder.
- Teachers will need to print out the "Search Comparison Worksheet" for each student.

Summary

Students investigate data organization, simulate linear and binary searches, and write pseudocode and Python for linear and binary search methods.

Outcomes

- Students will recognize the differences between linear search and binary search and will be able to recognize which method is suitable for a given problem.
- Students will be able to code both binary and linear searches.

Overview

Session 1

1. Getting Started (5 min) - Journal
2. Class Discussion and Activities (25 min) - Introduction to linear and binary search algorithms
3. Coding Linear Search (20 min)

Session 2

1. Getting Started (5 min) - Think-Pair-Share
2. Coding Activity (30 min) - Write pseudocode and implement binary search
3. Compare Searches (15 min) - Fill out worksheet to compare search algorithms

Source

Phone book presentation adapted from a lesson taught by Dr. Rheingans in CMSC 201 at the University of Maryland, Baltimore County

Learning Objectives

CSP Objectives

- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4D - Effective collaboration strategies enhance performance.
- 1.2.5 - Analyze the correctness, usability, functionality, and suitability of computational artifacts.
  - 1.2.5A - The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
  - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
  - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
  - 4.1.1E - Algorithms can be combined to make new algorithms.
  - 4.1.1F - Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.
  - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
  - 4.1.1H - Different algorithms can be developed to solve the same problem.
  - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
  - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
  - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
  - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
  - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
- 4.2.1 - Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time.
  - 4.2.1A - Many problems can be solved in a reasonable time.
  - 4.2.1B - Reasonable time means that the number of steps the algorithm takes is less than or equal to a polynomial function (constant, linear, square, cube, etc.) of the size of the input.
  - 4.2.1C - Some problems cannot be solved in a reasonable time, even for small input sizes.
- 4.2.4 - Evaluate algorithms analytically and empirically for efficiency, correctness and clarity.
  - 4.2.4A - Determining an algorithm's efficiency is done by reasoning formally or mathematically about the algorithm.
  - 4.2.4B - Empirical analysis of an algorithm is done by implementing the algorithm and running it on different inputs.
  - 4.2.4C - The correctness of an algorithm is determined by reasoning formally or mathematically about the algorithm, not by testing an implementation of the algorithm.
  - 4.2.4D - Different correct algorithms for the same problem can have different efficiencies.
  - 4.2.4E - Sometimes, more efficient algorithms are more complex.
  - 4.2.4F - Finding an efficient algorithm for a problem can help solve larger instances of the problem.
  - 4.2.4H - Linear search can be used when searching for an item in any list; binary search can be used only when the list is sorted.
- 5.1.2 - Develop a correct program to solve problems.
  - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
  - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
  - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
  - 5.1.2D - Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.
  - 5.1.2E - Documentation about program components, such as code segments and procedures, helps in developing and maintaining programs.
  - 5.1.2F - Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments.
  - 5.1.2G - Program development includes identifying programmer and user concerns that affect the solution to problems.
- 5.1.3 - Collaborate to develop a program.
  - 5.1.3B - Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.
  - 5.1.3C - Collaboration in the iterative development of a program requires different skills than developing a program alone.
  - 5.1.3F - Effective communication between participants is required for successful collaboration when developing programs.
- 5.3.1 - Use abstraction to manage complexity in programs.
  - 5.3.1A - Procedures are reusable programming abstractions.
  - 5.3.1B - A procedure is a named grouping of programming instructions.
  - 5.3.1C - Procedures reduce the complexity of writing and maintaining programs.
  - 5.3.1D - Procedures have names and may have parameters and return values.
  - 5.3.1E - Parameterization can generalize a specific solution.
  - 5.3.1F - Parameters generalize a solution by allowing a procedure to be used instead of duplicated code.
  - 5.3.1G - Parameters provide different values as input to procedures when they are called in a program.
  - 5.3.1L - Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
  - 5.5.1A - Numbers and numerical concepts are fundamental to programming.

- 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
- 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
- 5.5.1F - Compound expressions using and, or, and not are part of most programming languages.
- 5.5.1H - Computational methods may use lists and collections to solve problems.
- 5.5.1J - Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection.

4.1.1 GH

4.1.2 AFI

CSTA K-12 Computer Science Standards

Collaboration

- 2-4: Exhibit dispositions necessary for collaboration: providing useful feedback, integrating feedback, understanding and accepting multiple perspectives, socialization.

Computational Thinking

- 2-12. Use abstraction to decompose a problem into sub-problems.
- 2-6. Describe and analyze a sequence of instructions being followed.
- 3B-4. Evaluate algorithms by their efficiency, correctness, and clarity

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.

Common Core Math:

- F-BF.1-2: Build a function that models a relationship between two quantities

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure

NGSS Practices:

- 2. Developing and using models
- 3. Planning and carrying out investigations
- 5. Using mathematics and computational thinking

Key Concepts

Students will:

- Understand the definition of a search.
- Be able to identify how the order of data influences which methods are appropriate for searching the data.
- Be able to describe linear and binary search algorithms in pseudocode and in Python.
- Understand the concept of *efficiency* when searching for an item.

Essential Questions

- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How can computational models and simulations help generate new understanding and knowledge?
- How can computation be employed to help people process data and information to gain insight and knowledge?
- What considerations and trade-offs arise in the computational manipulation of data?
- What opportunities do large data sets provide for solving problems and creating knowledge?
- How are algorithms implemented and executed on computers and computational devices?
- What kinds of problems are easy, what kinds are difficult, and what kinds are impossible to solve algorithmically?
- How are algorithms evaluated?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

In Lesson Resources Folder:

- Search Comparison Worksheet to use at the end of Session 2
- DataSets folder that contains the six datasets for the Search Comparison Worksheet
- SearchCode.py contains skeleton code for a numeric search program

Lesson Plan

Session 1

Getting Started (5 min)

Journal: What would be the best way to organize a collection of DVDs so that you could find the one you want very quickly? Discuss.

Class Discussion and Activities (25 min)

Linear Search Discussion (5 min)

As a class, discuss the following question: What is the most effective way to look for an item in an unordered set of values?

Possible Answers:

- Randomly (How do you stop yourself from repeating yourself?)
- One-by-one

Linear Search Activity (5 min)

1. Pass out pieces of paper with numbers on them to everyone in the class except one person. Students should not share their numbers with anyone.
2. Have everyone stand at the front of the room in a line. The person without a number stands in front and is assigned a number to look for. The class should keep track of how many people they have to ask before they get the number (students that have been checked should sit down).
    - Do this activity a few times. In between, each student should switch with another student a couple of times without showing their paper. (Note: A fun way to do this might be a snowball fight if you can)
    - Ask for a number that does not exist in the set at least once. What happens? How many people does it take before they figure out the number is not in the set?
3. Have everyone sit down but keep their papers.

Binary Search Discussion and Presentation (10 min)

Take out a dictionary (or phone book). Ask the class, how would you search for a particular word/name?

Steps for Binary Search in a book of items to demonstrate to the class:

1. Flip to the middle and pick a word in the middle of the page
2. Is your word higher or lower than this word? If it is higher, “throw out” the lower half of the book. If it is lower, “throw out” the top half. (Not literally unless it is a very old phonebook. Students do love it when you tear up the phonebook, though, and it makes for a very effective demonstration.)
3. Repeat steps 1 and 2 until you find the word.

Why would this not work for an unordered list?

Binary Search Activity (5 min)

1. Have everyone go to the front of the room and get in numerical order by paper. Repeat the search activity using the binary search algorithm.
2. Try with a number not in the list. How can you tell when it doesn’t exist?

Coding Linear Search (20 min)

1. Have students work in pairs to write a code for linear search. The code should:
    1. Read in a csv file that has a list of unordered numbers. (They should input the file name from the user.)
    2. Ask the user what number they want to find and validate that number.
    3. Tell the user whether the number was found. If it was, it should output the number of items it had to look at.
2. Students should save their code for the next day.

Note: Skeleton code (SearchCode.py) is provided in the Lesson Resources Folder.

Session 2

Getting Started (5 min)

Think-Pair-Share

- Ask students to list non-numbered, real-world things that they search for or sort/order in their daily lives.
- Can all data be sorted, or do types of data exist that cannot be sorted? How would you organize and search these types of data?

Coding Activity (30 min)

Part 1 Pseudocode (10 min)

1. As a class, write down the steps for Binary Search on the board.
2. In pairs, have the students write pseudocode for how they would implement binary search.

Part 2 Coding (20 min)

1. Students should use their pseudocode to write a program for binary search. (They can also make use of the skeleton code provided for linear search.) The code should accomplish the same things as linear search: read in a file, get a number, and output if the number is found and how many items were checked.

Comparing Searches (15 min) – (May also be Homework if programs are not finished)

- Pass out the worksheet "Search Comparison Worksheet" from the lesson resources folder.
- Students should run both their linear and binary search programs with the six provided datasets of increasing sizes, (also in the lesson resources folder.)
- As they go through, students should record their results in the worksheet and answer the questions at the bottom.
- Discuss the results as a class.

Options for Differentiated Instruction

For students that have difficulty understanding the concepts of searching for items in a set of data, pair those students with a student who has a firm grasp of the concept for the activities. Have the pair work together for 1A and then have them keep their own paper secure using the extra game sheet (1A'). Simiilarly for 1B - 1B' and 1C - 1C'.

Evidence of Learning

Formative Assessment

Correctness of Python functions for linear search and binary search

Summative Assessment

"Searching Assessment Items.docx" in lesson folder

"Search Comparison Worksheet" in the lesson folder

Internal Use Only

Pending Tasks

~~Remove battleship exercise (replace intro with our own "standing up and searching" student "simulation" exercise)~~

~~Session 1: introduce concepts, student simulation, create test data (sorted and unsorted), design and implement linear search (output: whether the item was found, what location it was in, how much items were tested)~~

~~Session 2: design and implement binary search, use provided test data sets (three of different sizes -- 100, 1000, 10000 numbers; sorted and unsorted) - compare output and discuss. Write short report for homewe~~rk.

(http://csmatters.org) 5 - 3

0b101 - 0b11

Sorting

Unit 5. Data Manipulation

Revision Date: Jul 20, 2016 (Version 2.0)
Duration: 3 50-minute sessions

Lesson Summary

Summary

In this three-session lesson, students explore and confront the difficulties of the problem of sorting data and the difficulties involved in expressing a clear and efficient algorithm for sorting.

Outcomes

- Students will be able to relate a real-world task such as sorting cards to sorting/organizing information in a computer.
- Students will understand the problem of sorting and why it is nontrivial for large data sets.
- Students will be able to describe in pseudocode a simple sorting algorithms (bubblesort).
- Students will be able to reason about the correctness and efficiency of different sorting algorithms, and will understand that the time required to sort a data set increases as the size of the data set grows.

Overview

Session 1:

1. Getting Started (5 min) - Journal
2. Activity Card Sorting (40 min)
    1. Explain the Problem [10 min]
    2. Paired Activity: Algorithm Creation [30 min]
3. Wrap Up (5 min)

Session 2:

1. Getting Started (5 min) - Journal
2. Class Discussion (5 min)
3. Group Activity: Algorithm Evaluation (15 min)
4. Group Activity: Algorithm Selection and Justification (20 min)
5. Wrap Up (5 min)

Session 3:

1. Getting Started (10 min) - Journal and short discussion
2. Guided Activity: Algorithm Analysis (10 min)
3. Paired Activity: Algorithm Analysis (20 min)
4. Wrap Up (10 min)

Learning Objectives

CSP Objectives

- 1.2.5 - Analyze the correctness, usability, functionality, and suitability of computational artifacts.
    - 1.2.5A - The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact.
    - 1.2.5B - A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact.
    - 1.2.5C - The functionality of a computational artifact may be related to how it is used or perceived.
    - 1.2.5D - The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.
- 2.2.1 - Develop an abstraction when writing a program or creating other computational artifacts.
    - 2.2.1B - An abstraction extracts common features from specific examples in order to generalize concepts.
- 2.3.1 - Use models and simulations to represent phenomena.

    - 2.3.1A - Models and simulations are simplified representations of more complex objects or phenomena.
- 4.1.1 - Develop an algorithm for implementation in a program.
    - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
    - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
    - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
    - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
    - 4.1.1E - Algorithms can be combined to make new algorithms.
    - 4.1.1F - Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.
    - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
    - 4.1.1H - Different algorithms can be developed to solve the same problem.
    - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
    - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
    - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
    - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
    - 4.1.2D - Different languages are better suited for expressing different algorithms.
    - 4.1.2F - The language used to express an algorithm can affect characteristics such as clarity or readability but not whether an algorithmic solution exists.
    - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
    - 4.1.2I - Clarity and readability are important considerations when expressing an algorithm in a language.
- 4.2.1 - Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time.
    - 4.2.1A - Many problems can be solved in a reasonable time.
    - 4.2.1B - Reasonable time means that the number of steps the algorithm takes is less than or equal to a polynomial function (constant, linear, square, cube, etc.) of the size of the input.
- 4.2.4 - Evaluate algorithms analytically and empirically for efficiency, correctness and clarity.
    - 4.2.4A - Determining an algorithm's efficiency is done by reasoning formally or mathematically about the algorithm.
    - 4.2.4B - Empirical analysis of an algorithm is done by implementing the algorithm and running it on different inputs.
    - 4.2.4C - The correctness of an algorithm is determined by reasoning formally or mathematically about the algorithm, not by testing an implementation of the algorithm.
    - 4.2.4D - Different correct algorithms for the same problem can have different efficiencies.
    - 4.2.4E - Sometimes, more efficient algorithms are more complex.
    - 4.2.4F - Finding an efficient algorithm for a problem can help solve larger instances of the problem.
    - 4.2.4H - Linear search can be used when searching for an item in any list; binary search can be used only when the list is sorted.
- 5.1.3 - Collaborate to develop a program.
    - 5.1.3A - Collaboration can decrease the size and complexity of tasks required of individual programmers.
    - 5.1.3B - Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.
    - 5.1.3C - Collaboration in the iterative development of a program requires different skills than developing a program alone.
    - 5.1.3D - Collaboration can make it easier to find and correct errors when developing programs.
    - 5.1.3F - Effective communication between participants is required for successful collaboration when developing programs.

2.2.1 B

2.3.1 A

4.1.1 G

4.1.2 I

Math Common Core Practice:

- MP2: Reason abstractly and quantitatively.
- MP4: Model with mathematics.
- MP8: Look for and express regularity in repeated reasoning.

Key Concepts

The algorithmic techniques and analysis involved in sorting data are seen in a wide variety of contexts and applications. Sorting numbers in a list is challenging but foundational to many algorithms in computer science.

Essential Questions

- How can computation be employed to facilitate exploration and discovery when working with data?
- What considerations and trade-offs arise in the computational manipulation of data?
- Why are some languages better than others when used to implement algorithms?
- What kinds of problems are easy, what kinds are difficult, and what kinds are impossible to solve algorithmically?
- How are algorithms evaluated?

- Which mathematical and logical concepts are fundamental to computer programming?

What makes a "good" algorithm?

What should be taken into consideration when comparing algorithms that complete the same task?

Teacher Resources

Student computer usage for this lesson is: **optional**

- You will need enough playing cards for every pair of students to have eight different cards. Alternatively, you may make your own cards (e.g., using index cards) with different numbers on them in place of playing cards.
- Handout - Sorting Algorithm Evaluation.docx (in lesson folder)
- Video collection - https://www.youtube.com/user/AlgoRythmics/videos (https://www.youtube.com/user/AlgoRythmics/videos)
  - **Note:** If students do not have access to computers to individually watch the sorting videos during the paired activity in Session 3, you could instead choose one of the algorithms and show it to the class, having all pairs complete the algorithm evaluation handout for that selected method.

Lesson Plan

Session 1

Getting Started (5 min)

Journal: Have students respond to the following questions:

- If you had 1 million books, and you had to be able to find any book by its title as fast as possible, how would you organize them?
- How many books would you need to look at in the worst case scenario to find the title *before* you have organized the books?
- How many books would you need to look at in the worst case scenario to find the title *after* you have organized the books?

Teacher note: Having just finished the lessons on searching, students should recall that searching an ordered list is faster than searching an unordered list.

Activity: Card Sorting (40 min)

Teacher note: The focus should be directed more toward the problem-solving technique than nitpicking about the language used. Although students are writing instructions for a human to manipulate a set of playing cards, they still need to be precise, because the assumption is that the person doesn't know what they are doing. This problem is challenging and will require creativity.

Explain the Problem [10 min]

- Demonstrate the card sorting sorting task as you explain.
- Clarify the goal: Today, you and a partner are going to design an algorithm and list the instructions for a person to arrange a row of playing cards into order (from lowest to highest value).
- Explain the basic rules:
  - If a card is on the table, it must be face down.
  - You can only see the value of a card by picking it up and looking at its face.
  - You can only be holding and looking at two cards at a time (1 in each hand).
  - You can compare the values of any of the cards you are holding in your hands and determine if one is greater, less than, or equal to the other card.
  - When you put a card down, try to be clear about *where* it should be put back down. Cards should be put face down.
  - You cannot use your memory of face-down cards to make decisions about them. You should behave as though you have no recollection of cards that you aren't currently holding.
  - You will have eight cards to practice, but the procedure you follow should be general enough to work for *any number* of cards.
- Ask the students whether there are any questions about the rules.
- Emphasize to students that there are many ways to achieve this task. As a class, they should try to come up with as many different ways of sorting as possible.

Paired Activity: Algorithm Creation [30 min]

- If there is an odd number of students, there could be one group of three students.
- Distribute the cards to student pairs. The cards can be ordinary playing cards, but each pair should receive cards from the same suit that have been shuffled. Alternatively, you can use handmade cards with arbitrary numbers on them. Before starting, have students agree on the ordering of the cards (e.g., whether aces are high or low).
- Have students write their instruction list (algorithm) on a piece of paper.

- The format of the instruction lists is up to the students; they can create a numbered list, a flow chart, a diagram with text and arrows, or other means of communication.
- Students should be working productively for the rest of the class meeting, designing and writing their card sorting algorithm.
- Circulate around the room to make sure that students are on task and that they understand the rules and goals of the activity.

Wrap Up (5 min)

Journal:

- If you were to give your algorithm a name that describes how it sorts, what would you name it?
- Identify the most difficult part of writing down the instructions for your algorithm.

Homework: Any pairs that did not finish the activity should complete it as a homework assignment before the next session.

Session 2

In this session, students review the sorting algorithms they wrote in session 1. Students will follow the algorithms created by their classmates and discover a variety of sorting strategies. By analyzing the various algorithms, students will attempt to find the "best" sorting strategy.

Teacher note: There are two main difficulties in algorithm design to highlight: (1) It is very difficult to be precise with language without some agreement about what terms mean. (2) Solving the problem by determining the strategies and steps required to sort objects correctly, as well as efficiently, presents a second level of difficulty.

Getting Started (5 min)

Journal: How do you think a sorting algorithm should be "measured" to determine if it is the "best"?

Class Discussion (5 min)

Pose the following question to the students: “In order to choose the best algorithm, we need to be able to measure each algorithm. What actions do you think we should count?” Give students an opportunity to respond individually and collectively. Optionally, you may use a think-pair-share approach or small groups to develop ideas and then share with the class.

Group Activity: Algorithm Evaluation (15 min)

- Distribute the "Sorting Algorithm Evaluation.doc" worksheet (in the lesson folder).
- Re-distribute playing cards to student pairs.
- Instruct students to follow directions on the Sorting Algorithm Evaluation worksheet and use it to record their experiences.

Group Activity: Algorithm Selection and Justification (20 min)

Create groups of four by joining the pairs who previously exchanged algorithms.

Teacher note: This "swapping algorithm" activity works especially well when students exchange algorithms with a group that has a fundamentally different approach. However, as a practical matter, this can be hard to arrange. From your observations during Session 1, you might have a sense of groups with different approaches that you can assign to swap algorithms.

- Groups should give feedback to each other about the algorithm, making sure to discuss:
  - The algorithm's correctness - does it work; do you understand it; could you simulate it or act it out?
  - Explain confusing/ambiguous parts of the other group's algorithm in order to help them write it better.
  - Discuss which of the two algorithms is "better" and be able to explain why.
  - Each group of 4 will nominate one of the two algorithms as the better one of the group.
- Ask each group to volunteer the better algorithm at their table.
- Act out/simulate one group's instructions.

Teacher note: It is possible that the nominated algorithm won't work perfectly. If you encounter any problems with the directions, give them the benefit of the doubt and simulate it as best you can to enable the class to understand the intent.

- Ask if other groups solved the problem with a different strategy and demonstrate a few groups' algorithms.
- Engage students in a discussion about which algorithm was the best. Why is it best? How should algorithms be evaluated?
- Ask students to argue for one method or another and explain their reasons.

Teacher note: The students will perform an actual analysis in a later lesson, so it's okay at this point to simply guide the discussion to see how students are thinking. They will re-examine these ideas later.

- Point out to the students that the speed at which the actor can follow the algorithm is not a measure of the algorithm, but rather the speed of the person. As an analogy, you might compare a person's walking speed with the distance they have to travel. To compare algorithms, you need to measure some "units of work". What those units are is debatable, but must to be agreed upon. For card sorting, "work" could mean picking a card up, putting it down, comparing with another card, etc.

Wrap Up (5 min)

Remind students of the two main issues in writing effective algorithms:

- The need for a clear, unambiguous language for expressing algorithmic solutions.

- Defining criteria for determining whether an algorithm is "good."

Homework: Assign students to write a final version of their algorithm, working out any ambiguities or other problems revealed during the activities.

Session 3

In this session, we end the set of sorting activities by relating sorting to algorithms in the real world. A further exploration of algorithm analysis with some new algorithms will sharpen their intuition about what should and shouldn't be "counted" when analyzing algorithms, what is "hard" for a computer to do, or what takes a "long time."

Getting Started (10 min)

Journal: discuss ideas and elements from the previous lesson:

- What is "work" for a computer?
- Why are algorithms measured the way they are?

Teacher should clarify that in general, we want two things from an algorithm:

- To provide a correct solution for any given input.
- To use computational resources as efficiently as possible.

Guided Activity: Algorithm Analysis (10 min)

- Go to the website that shows folk dancers simulating various sorting algorithms. https://www.youtube.com/user/AlgoRythmics/videos (https://www.youtube.com/user/AlgoRythmics/videos)
- With the students, click on the bubble sort video, and complete Sorting Algorithm Evaluation.docx for bubble sort, helping the students to identify when a "comparison" is occurring and when a "swap" is occurring.
- Play it again and help the students write the pseudocode for the bubble sort algorithm.

Paired Activity: Algorithm Analysis (20 min)

- Assign each pair a different sorting algorithm.
- Have each pair watch their assigned sorting video and complete the Sorting Algorithm Evaluation for that method.
- Have each group attempt to write pseudocode for their assigned sorting algorithm. Some of the algorithms are quite complex, so emphasize to the students that the goal of this exercise should be to think about what's happening, rather than to get it completely "right."

Wrap Up (10 min)

- Instruct students to discuss the following prompts with an elbow partner and then, collaboratively write a response in their journals that incorporates the ideas of both partners.
  - What should and shouldn't be "counted" when analyzing algorithms?
  - What is "hard" or time consuming for a computer to do?
  - Why is the efficiency of algorithms important?
- Assign homework for next lesson on comparing algorithms (Unit 5, Lesson 4): Identify in your journal two places that you often travel between. Of the alternative routes available, what do you consider to be the best route? Why? Are there circumstances in which an alternate route is better? When is that the case?

Options for Differentiated Instruction

Suggestion: If you have a mix of new and advanced students, challenge the advanced students to sort twice as many cards with a parallel processing algorithm of their own design. Each student on the team can perform one action at the same time.

Evidence of Learning

Formative Assessment

Evaluation of algorithms

Convert actions into an algorithm

Internal Use Only

Future Work

It would be helpful to offer some specific ideas for summative assessment.

(http://csmatters.org) 5 - 4

0b101 - 0b100

Comparing Algorithms

Unit 5. Data Manipulation

Revision Date: Jul 18, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Pre-lesson Preparation: You should familiarize yourself with www.sorting-algorithms.com/ (http://www.sorting-algorithms.com/,) paying particular attention to the variety of algorithms and settings along the top of the page. For session 2, you should have the timedsorts.py code and data files (in the lesson folder) readily available for your students.

Summary

In this two-session lesson, students will explore algorithmic efficiency. They will understand the idea through discussion, manual analysis of simple algorithms, and data collection for implemented algorithms.

Outcomes

Students will be able to:

- identify algorithms that have different efficiencies in their problem solving approach.
- explain the metrics used to describe efficiency.
- perform an empirical analysis of sorting algorithms by running the algorithms on different inputs.

Overview

Session 1:

1. Getting Started (5 min)
2. Guided Activity (40 min)
    1. Good Algorithms and Better Algorithms (5 min)
    2. Algorithmic Efficiency (10 min)
    3. Computational Complexity (10 min)
    4. Comparing Sorting Algorithms (15 min)
3. Wrap Up (5 min)

Session 2:

1. Getting Started (5 min)
2. Empirical Investigation (40 min)
    1. Introduction (5 min)
    2. Experimental Design (10 min)
    3. Data Collection (25 min)
3. Wrap Up (5 min)

Learning Objectives

CSP Objectives

- 1.2.5 - Analyze the correctness, usability, functionality, and suitability of computational artifacts.
  - 1.2.5A - The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact.
  - 1.2.5B - A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact.
  - 1.2.5C - The functionality of a computational artifact may be related to how it is used or perceived.
  - 1.2.5D - The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.
- 2.2.2 - Use multiple levels of abstraction to write programs.
- 2.2.3 - Identify multiple levels of abstractions used when writing programs.
- 2.3.1 - Use models and simulations to represent phenomena.
- 2.3.2 - Use models and simulations to formulate, refine and test hypotheses.
- 3.1.1 - Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.
- 3.1.2 - Collaborate when processing information to gain insight and knowledge.
- 3.1.3 - Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.
- 4.2.1 - Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time.
  - 4.2.1A - Many problems can be solved in a reasonable time.
  - 4.2.1B - Reasonable time means that the number of steps the algorithm takes is less than or equal to a polynomial function (constant, linear, square, cube, etc.) of the size of the input.
  - 4.2.1C - Some problems cannot be solved in a reasonable time, even for small input sizes.
- 4.2.4 - Evaluate algorithms analytically and empirically for efficiency, correctness and clarity.
  - 4.2.4A - Determining an algorithm's efficiency is done by reasoning formally or mathematically about the algorithm.
  - 4.2.4B - Empirical analysis of an algorithm is done by implementing the algorithm and running it on different inputs.
  - 4.2.4C - The correctness of an algorithm is determined by reasoning formally or mathematically about the algorithm, not by testing an implementation of the algorithm.
  - 4.2.4D - Different correct algorithms for the same problem can have different efficiencies.
  - 4.2.4E - Sometimes, more efficient algorithms are more complex.
  - 4.2.4F - Finding an efficient algorithm for a problem can help solve larger instances of the problem.
  - 4.2.4G - Efficiency includes both execution time and memory usage.

4.2.1 AB

4.2.4 ABDE

Math Common Core Practice:

- MP5: Use appropriate tools strategically.

Common Core Math:

- S-ID.1-4: Summarize, represent, and interpret data on a single count or measurement variable
- S-ID.5-6: Summarize, represent, and interpret data on two categorical and quantitative variables

Common Core ELA:

- WHST 12.1 - Write arguments on discipline specific content
- WHST 12.4 - Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products
- WHST 12.7 - Conduct short as well as more sustained research projects to answer a question

NGSS Practices:

- 1. Asking questions (for science) and defining problems (for engineering)
- 2. Developing and using models
- 3. Planning and carrying out investigations
- 4. Analyzing and interpreting data
- 8. Obtaining, evaluation, and communicating information

Essential Questions

- How can computational models and simulations help generate new understanding and knowledge?
- What kinds of problems are easy, what kinds are difficult, and what kinds are impossible to solve algorithmically?
- How are algorithms evaluated?

Teacher Resources

Student computer usage for this lesson is: **required**

Sorting:

- Python code for bubble sort and other sorting methods (for use in Session 1 guided activity):
  - http://interactivepython.org/runestone/static/pythonds/SortSearch/sorting.html (http://interactivepython.org/runestone/static/pythonds/SortSearch/sorting.html)
- Sorting Algorithms Animation timing comparison tool (for use in Session 2 guided activity):
  - http://www.sorting-algorithms.com/ (http://www.sorting-algorithms.com/)
- Many online resources are available to help in understanding different sort algorithms.
  - 15 Sorting Algorithms in 6 minutes
    - https://www.youtube.com/watch?v=kPRA0W1kECg (https://www.youtube.com/watch?v=kPRA0W1kECg)
  - Bubble Sort versus Quick Sort
    - https://www.youtube.com/watch?annotation_id=annotation_3243502817&feature=iv&src_vid=92WHN-pAFCs&v=aXXWXz5rF64 (https://www.youtube.com/watch?annotation_id=annotation_3243502817&feature=iv&src_vid=92WHN-pAFCs&v=aXXWXz5rF64)
  - Merge Sort versus Quick Sort
    - https://www.youtube.com/watch?annotation_id=annotation_492880&feature=iv&src_vid=aXXWXz5rF64&v=es2T6KY45cA (https://www.youtube.com/watch?annotation_id=annotation_492880&feature=iv&src_vid=aXXWXz5rF64&v=es2T6KY45cA)
  - Merge Sort in more detail
    - http://www.zutopedia.com/ms_vs_qs.html (http://www.zutopedia.com/ms_vs_qs.html)
  - Classic “Sorting Out Sorting” four parts
    - https://www.youtube.com/watch?v=YvTW7341kpA (https://www.youtube.com/watch?v=YvTW7341kpA)
    - https://www.youtube.com/watch?v=plAi7kcqMNU (https://www.youtube.com/watch?v=plAi7kcqMNU)
    - https://www.youtube.com/watch?v=gtdfW3TbeYY (https://www.youtube.com/watch?v=gtdfW3TbeYY)
    - https://www.youtube.com/watch?v=wdcoRfS8edM (https://www.youtube.com/watch?v=wdcoRfS8edM)
  - Sorting Animations
    - https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html (https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html)
  - Comparing three sorts
    - http://vinayakgarg.wordpress.com/2011/10/25/time-comparison-of-quick-sort-insertion-sort-and-bubble-sort/ (http://vinayakgarg.wordpress.com/2011/10/25/time-comparison-of-quick-sort-insertion-sort-and-bubble-sort/)
  - Comparison of Sorting Algorithms Approach
    - http://warp.povusers.org/SortComparison/index.html (http://warp.povusers.org/SortComparison/index.html)
  - Integers
    - http://warp.povusers.org/SortComparison/integers.html (http://warp.povusers.org/SortComparison/integers.html)
  - Integers with High Repetition
    - http://warp.povusers.org/SortComparison/integers_rep.html (http://warp.povusers.org/SortComparison/integers_rep.html)
  - Strings (slow compare fast move)
    - http://warp.povusers.org/SortComparison/strings.html (http://warp.povusers.org/SortComparison/strings.html)
  - Arrays (fast compare slow copying)
    - http://warp.povusers.org/SortComparison/arrays.html (http://warp.povusers.org/SortComparison/arrays.html)

Lesson Plan

Session 1

Getting Started (5 min)

Think-Pair-Share: Alternate Routes

- If you assigned the homework from the previous lesson, ask your students to get out their journals to discuss their entries. If not, you could have them write a response in their journal to the following prompt:
  - Identify two places that you often travel between. Of the alternative routes available, what do you consider to be the best route? Why? Are there circumstances in which an alternate route is better? When is that the case?
- Have your students pair off to discuss their responses for a minute or two.
- Ask some of the pairs to share and summarize their journal entries.

Guided Activity (40 min)

Good Algorithms and Better Algorithms [5 min]

Briefly discuss with your class the topic: what properties make for a good algorithm? What makes one algorithm better than another? Properties you may want to discuss if your students do not volunteer them:

- correctness
- ease of understanding
- elegance (clarity, simplicity, and inventiveness)
- efficiency

A good analogy is purchasing a car, where people are concerned about:

- safety
- ease of handling
- style
- fuel efficiency

Today's session will address the topic of efficiency.

Algorithmic Efficiency [10 min]

Introduce the concept of algorithmic efficiency to your students by asking them if any can describe what algorithmic efficiency is, or what it means for an algorithm to be efficient. Briefly describe efficiency as how well an algorithm uses two resources, time and space (stored memory), to solve a problem. Some topics you may wish to discuss include:

- Two algorithms may both solve the same problem correctly, but with different degrees of efficiency.
- An algorithm that is maximally efficient will minimize the resources it uses.
- Algorithms typically face a space-time tradeoff, where they either use more memory to run faster or take more time but use less memory.
    - When you use a map, you are using more storage resources to go along your route more quickly
- An example of an algorithm that trades space for time (stores more in memory to operate faster) is a lookup table.
    - A real-world example of a lookup table is numbered valet parking. The valet gives the customer a number and goes to park the customer's vehicle in the parking space with that number. When the customer or valet needs to find the vehicle again, instead of having to search through all the spaces, all they need is the remembered (stored) number to go directly to that parking space.
- Most of the time, we are more interested in *computational efficiency*, or time usage of an algorithm.

Computational Complexity [10 min]

Teacher note: This topic is more advanced, so you may wish to go more in depth or move on to the activity, as appropriate for your students.

A central idea of algorithms is that some algorithms will take more and more time as the size of their input increases. Time is not measured in seconds but rather the number of computational steps needed for the algorithm to finish operation on a given input. Great algorithms grow linearly, at the same rate as their input, meaning the time it takes to finish is directly proportional to the size of the problem they are solving (amount of input data). For instance, an algorithm that takes 10 steps for an input of size 10 and 1000 steps for an input of size 1000 is said to be linear in its input. However, most algorithms take longer as their input gets larger. For instance, an algorithm that takes only 25 steps for an input of size 5 may take 100 steps for an input of size 10, 10000 steps for an input of size 100, and one million steps for a size of only 1000 (it is taking quadratically more time as the input gets larger).

When we analyze algorithms, we often talk about the algorithm's *computational complexity*, which is the order of magnitude of the algorithm's running time. We almost always discuss the worst case complexity, since that is a bound on the resources required.

If an algorithm finishes with the same number of steps regardless of the size of its input, it is called *constant time*, which is O(1) in mathematical form (read aloud as "big-oh one"). Constant time algorithms are the fastest in terms of computational efficiency, and any algorithm that takes a constant number of steps is considered O(1). An algorithm that takes 10 steps for an input of size 10 and also takes 10 steps for an input of size 1000 is likely O(1). However, very, very few algorithms are constant time because most algorithms necessarily take longer as the size of their input increases.

An algorithm that can finish by looking at each piece of its input only once is called *linear time* or *linear order*, and is written mathematically as O(n), where n stands for "the size of the input." An algorithm that takes 10 steps for an input of size 10 and also takes 1000 steps for an input of size 1000 is likely O(n). Very few algorithms are linear order, especially if they must compare pieces in their input, such as sorting algorithms. The best sorting algorithms are somewhere between linear time and quadratic *polynomial time*, written as $O(n^2)$, where n^2 stands for "the size of the input, squared." Any algorithm that is $O(n^2)$ typically must compare each piece of its input with every other piece of input at least once. An algorithm that takes 100 steps for input of size 10 and a million steps for input of size 1000 is likely $O(n^2)$.

Most sorting algorithms are of an order between O(n) and $O(n^2)$ known as *linearithmic time*, written as O(n log n), where log is the logarithmic function. In fact, O(n log n) is the fastest possible order for a comparison-based sorting algorithms. It is impossible for such algorithms to be O(n) since they must make at least some comparisons of their input data.

Comparing Sorting Algorithms [15 min]

Using the simulation tools at http (http://www.sorting-algorithms.com/)://www.sorting-algorithms.com (http://www.sorting-algorithms.com/)/ (http://www.sorting-algorithms.com/), students will investigate, compare, and contrast sorting algorithms. Notice the grid in the center of the page. Each column is a particular sorting algorithm, and each row is an ordering of horizontal bars (either random, nearly sorted, reversed order, or few unique). Each algorithm will sort the bars in a given cell from top to bottom in increasing order by length.

Ask your students to interact with the website by clicking the green start icons and observing how long it takes each algorithm to sort its bars relative to the other algorithms.

Some questions to have them discuss or record in their journal could include:

- Find the row for "Random" and click the icon above it to see each algorithm sort a randomly ordered set of bars.
- Which algorithms are going slow on average? Which ones are fastest?
- Experiment with larger input sizes by clicking a number for Problem Size at the top (30, 40, or 50). Click the icon above Random again. What changes do you notice in the speeds of algorithms? Why are the slow algorithms taking even longer than before? Would you ever want to use them?
- Set the Problem Size back to 20

- Find the column for "Bubble" (Bubble sort) and click the icon above it to see it run on each of the ordering types. Which one finishes first? Why do you think that is?
- Find the row for "Nearly Sorted" and click the icon above it to see all the algorithms run on nearly sorted input data. Which algorithms finish first? What algorithm is slow on Random data but finishes quickly on Nearly Sorted data? Why do you think it does so?
- Which algorithms do you think are $O(n^2)$?

Make sure your students understand that the size and order of input data can affect how long an algorithm takes. You should direct or help your students discover that Bubble sort is a slow sorting algorithm that can be fairly fast for nearly sorted data. You may wish to discuss that Bubble sort is $O(n^2)$ in the worst case, explaining why it takes so long for large input, but is O(n) in the best case, which is when input is already (or nearly) sorted. In contrast, Selection sort is $O(n^2)$ in both the worst and best cases, and Merge sort is O(n log n) in both the worst and best cases. In general, most sorting algorithms that we would want to use are O(n log n), since $O(n^2)$ is usually too slow. You may also want to mention that Bubble sort is considered one of the most inefficient sorting algorithms and that Quick sort's worst performance is on already sorted data, so some Quick sort implementations shuffle the inputs before sorting to avoid that situation.

Wrapup (5 min)

Watch one or more of the available movie clips that compare the performance of sorting algorithms:

Suggested list of videos (Many more are available):

- 15 Sorting Algorithms in 6 minutes
  - https://www.youtube.com/watch?v=kPRA0W1kECg (https://www.youtube.com/watch?v=kPRA0W1kECg)
- Bubble Sort versus Quick Sort
  - https://www.youtube.com/watch?annotation_id=annotation_3243502817&feature=iv&src_vid=92WHN-pAFCs&v=aXXWXz5rF64 (https://www.youtube.com/watch?annotation_id=annotation_3243502817&feature=iv&src_vid=92WHN-pAFCs&v=aXXWXz5rF64)
- Merge Sort versus Quick Sort
  - https://www.youtube.com/watch?annotation_id=annotation_492880&feature=iv&src_vid=aXXWXz5rF64&v=es2T6KY45cA (https://www.youtube.com/watch?annotation_id=annotation_492880&feature=iv&src_vid=aXXWXz5rF64&v=es2T6KY45cA)
- Merge Sort in more detail
  - http (http://www.zutopedia.com/ms_vs_qs.html)://www.zutopedia.com/ms_vs_qs.html (http://www.zutopedia.com/ms_vs_qs.html)
- Classic "Sorting Out Sorting" in four parts
  - https://www.youtube.com/watch?v=YvTW7341kpA (https://www.youtube.com/watch?v=YvTW7341kpA)
  - https://www.youtube.com/watch?v=plAi7kcqMNU (https://www.youtube.com/watch?v=plAi7kcqMNU)
  - https://www.youtube.com/watch?v=gtdfW3TbeYY (https://www.youtube.com/watch?v=gtdfW3TbeYY)
  - https://www.youtube.com/watch?v=wdcoRfS8edM (https://www.youtube.com/watch?v=wdcoRfS8edM)

Session 2

Getting Started (5 min)

Journal: Remind your students about the sorting algorithms from the previous session and have them answer the following questions:

- What are some ways in which one algorithm can be better than another, besides efficiency?
- Explain what algorithmic efficiency is by discussing two different sorting algorithms.

Guided Activity: Empirical Analysis (40 min)

Introduction [5 min]

The students will measure and analyze the effect of sorting set size on execution time for a given sorting algorithm using Python code. Using the timedsorts.py file in the lesson resources folder as a basis, the students will perform an experimental analysis to compare sorting algorithms by timing them on input data of different sizees. They will hypothesize, design and code their experiment, collect results, and write a report for homework.

The sorting functions available in the Python code include: quick sort, merge sort, selection sort, insertion sort, and bubble sort. For advanced students or classes may, you may wish to have them implement additional sorting algorithms.

The sample code includes helper functions to generate random data, to load data from a file, and to time sorting functions on the data. Example code for invoking these functions is included at the end of the file. You can remove this example code before sharing it with your students if you wish to emphasize the programming and critical thinking required to do this project.

Each student (or pair or group) needs their own copy of the Python code to modify for their experiments.

Experimental Design [15 min]

Students will compare sorting algorithms by timing them with Python code on input data of various sizes. Have your students (individually or in pairs) make a hypothesis about what will happen as the size of data input increases, answering the following questions:

- How can you determine which sorting algorithm is most efficient and which is least effiicient?
- What sorting algorithm do you think is most efficient, and which is least efficient?
- What do you hypothesize will happen to the time as the size of the data input increases?
- What is the independent variable in this experiment?
- What is the dependent variable?

Have your students write out a description of the steps they will take to perform the experiment.

Data Collection [20 min]

Have students modify their Python sorting code to implement the experimental steps they outlined. Students must:

- Time their sorting routines with different size sets of items to sort (e.g., 5000, 10000, 25000, 50000). Sample data files are available in the lesson resources folder, but students should use the provided helper function to generate arrays of random data, too.
- Record (write down) the size of each input array, the name of the sorting function, and the resulting time it took to sort the data for each algorithm/data combination they test.
- Discuss the results with another student or group. What patterns can be seen in the relationship between the amount of data and the time to run the program?

The data collection should be completed by the end of class, but students will continue to work on this activity by writing a report describing their results.

Wrapup (5 min)

Assign as homework to write a short report about the findings, making sure to:

- Write your hypothesis. How do your findings reflect your hypothesis?
- What algorithm or algorithms are most efficient? Why?
- What algorithm is least efficient? Why?
- What values did you use for your independent variable?
- Present the data you collected in a table and in a graph.
- What conclusions can you draw about sorting algorithms?
- Explain why algorithmic efficiency is important by discussing another problem (not sorting) where a correct but inefficient algorithm is unusable at larger input sizes.
- Pick two sorting algorithms you tested. Write a paragraph for each describing how it works, and one paragraph comparing the two algorithms explaining which is more efficient and why (you can do research and look at the Python code to figure out the reasons).

Students must complete a short research report on their sorting algorithm research procedure, results, and analysis of the results.

Options for Differentiated Instruction

The teacher may decide to have the students choose how they want to organize the empirical analysis effort. Alternatively, scaffolding with a worksheet or checklist could be used to guide the students through the data collection and analysis tasks.

Evidence of Learning

Formative Assessment

The following "Checks for Understanding" could be used to guide the students towards the three learning objectives:

Objective: SWBAT identify families of correct algorithms that have different efficiencies in their problem solving approach.

1. Students will pair-share what makes a good choice for the route taken to get from point A to point B.
2. Students will compare algorithms and explain why and when some are better than others in terms of efficiency.
3. Students will be able to identify and rank order the least efficient sorting algorithms in the simulations.

Objective: SWBAT demonstrate logical reasoning and metrics is used to describe an algorithm's efficiency.

1. Predict: Students will have seen sorting algorithms implemented as folk dances. Students will predict -- for their algorithm -- how adding additional dancers would increase the dance completion time.

Objective: SWBAT to perform empirical analysis of sorting algorithms by running the algorithms on different inputs.

1. Students will work in pairs to collect data on sorting execution times. The pairs will share their results with other groups to check for patterns before they write up their results.

Summative Assessment

Students will complete a short research report on their sorting algorithm research procedure, results, and analysis of the results.

(http://csmatters.org) 5 - 5

0b101 - 0b101

Advanced Algorithms

Unit 5. Data Manipulation

Revision Date: Jul 18, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary: Students are introduced to the theory of computation, computability, the halting problem, and advanced algorithms. In particular, they will learn about heuristic search used by artificial intelligence (AI) programs to play games.

Objective:

Students will be able to:

- define computation and some basic ideas of the theory of computation
- discuss computability and understand there are some things computers cannot solve
- explain the Halting Problem
- identify some advanced search algorithms
- understand how AI programs represent games with game trees
- understand how AI programs use uninformed and heuristic search algorithms to play games

Overview:

Session 1

1. Getting Started (10 min)
2. Guided Activity (35 min)
    1. Inverse Operations Activity [10 min]
    2. Computation [10 min]
    3. Computability [15 min]
3. Wrap Up (5 min) - Think-Pair-Share

Session 2

1. Getting Started (5 min)
2. Guided Activity (45 min)
    1. Search and Game Trees [15 min]
    2. Game-Playing AI [10 min]
    3. Types of Heuristic Search [10 min]
    4. Playing a Game with Heuristic Search [10 min]

Learning Objectives

CSP Objectives

- 4.2.1 - Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time.
    - 4.2.1A - Many problems can be solved in a reasonable time.
    - 4.2.1B - Reasonable time means that the number of steps the algorithm takes is less than or equal to a polynomial function (constant, linear, square, cube, etc.) of the size of the input.
    - 4.2.1C - Some problems cannot be solved in a reasonable time, even for small input sizes.
    - 4.2.1D - Some problems can be solved but not in a reasonable time. In these cases, heuristic approaches may be helpful to find solutions in reasonable time.
- 4.2.2 - Explain the difference between solvable and unsolvable problems in computer science.
    - 4.2.2A - A heuristic is a technique that may allow us to find an approximate solution when typical methods fail to find an exact solution.
    - 4.2.2B - Heuristics may be helpful for finding an approximate solution more quickly when exact methods are too slow.
    - 4.2.2C - Some optimization problems such as "find the best" or "find the smallest" cannot be solved in a reasonable time but approximations to the optimal solution can.
    - 4.2.2D - Some problems cannot be solved using any algorithm.
- 4.2.3 - Explain the existence of undecidable problems in computer science.

    - 4.2.3A - An undecidable problem may have instances that have an algorithmic solution, but there is no algorithmic solution that solves all instances of the problem.
    - 4.2.3B - A decidable problem is one in which an algorithm can be constructed to answer "yes" or "no" for all inputs (e.g., "Is the number even?").
    - 4.2.3C - An undecidable problem is one in which no algorithm can be constructed that always leads to a correct yes-or-no answer.
- 4.2.4 - Evaluate algorithms analytically and empirically for efficiency, correctness and clarity.
    - 4.2.4A - Determining an algorithm's efficiency is done by reasoning formally or mathematically about the algorithm.
    - 4.2.4D - Different correct algorithms for the same problem can have different efficiencies.
    - 4.2.4E - Sometimes, more efficient algorithms are more complex.
    - 4.2.4F - Finding an efficient algorithm for a problem can help solve larger instances of the problem.

4.2.1 ABCD

4.2.2 ABCD

4.2.3 ABC

Math Common Core Practice:

- MP2: Reason abstractly and quantitatively.
- MP4: Model with mathematics.

Common Core Math:

- N-RN.3: Use properties of rational and irrational numbers.
- F-BF.1-2: Build a function that models a relationship between two quantities
- F-BF.3-5: Build new functions from existing functions
- S-IC.1-2: Understand and evaluate random processes underlying statistical experiments
- S-CP.6-9: Use the rules of probability to compute probabilities of compound events in a uniform probability model
- S-MD.5-7: Use probability to evaluate outcomes of decisions

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure

NGSS Practices:

- 2. Developing and using models
- 6. Constructing explanations (for science) and designing solutions (engineering)

Essential Questions

- How can computational models and simulations help generate new understanding and knowledge?
- What considerations and trade-offs arise in the computational manipulation of data?
- What kinds of problems are easy, what kinds are difficult, and what kinds are impossible to solve algorithmically?
- How are algorithms evaluated?

Teacher Resources

Student computer usage for this lesson is: **required**

Links to videos and online tools as indicated in the lesson plan.

- http (http://www.sorting-algorithms.com/)://www.sorting-algorithms.com/ (http://www.sorting-algorithms.com/) for sorting algorithm review
- The Halting Problem https:// (https://www.youtube.com/watch?v=92WHN-pAFCs)www.youtube.com/watch?v=92WHN-pAFCs (https://www.youtube.com/watch?v=92WHN-pAFCs) (7:52)

Alternative instruction could include the Towers of Hanoi problem and discuss the algorithm for solving it. Some demonstrations are available here:

- http://illuminations.nctm.org/Activity.aspx?id=4195 (http://illuminations.nctm.org/Activity.aspx?id=4195)
- http://www.mazeworks.com/hanoi/ (http://www.mazeworks.com/hanoi/)

Lesson Plan

Session 1

Getting Started (10 min)

Think-Pair-Share: In pairs, think about and try to answer each of the following questions:

- Given y = 7x + 4 and x=3 what are the steps to find y?
- Given y = 7x + 4 and y=3 what are the steps to find x?
- Factor 81,927,497 and 81,927,499. Can you figure out the steps?
- Multiply 431 x 433 x 439. What are the steps?

Note: just give them a few minutes to try the factoring, but round them up to continue and discuss: which operations were much harder to perform than their inverse? Can you just invert the steps, and why or why not?

Guided Activities (35 min)

Inverse Operations Activity [10 min]

1. Convey the following concepts:
    a. Inverse arithmetic operations
        - add/subtract [x + 7 – 7 = x – 7 + 7 = x] ;
        - multiply/divide [x * 7 / 7 = x / 7 * 7 = x] ;
        - e^x /ln(x) [$\ln(e^x) = e^{\ln(x)} = x$];
        - …
    b. Some arithmetic operations are harder to do then their inverse operations -- *as the students did during the warm-up*.
        - Cubing a number versus finding the cube root of the result.
            - Find the cube of 12
            - Find the cube root of 5832
        - Multiplying numbers to form a product versus factoring the product.
    c. The same can be true with algorithms.
        - It is much easier to scramble a Rubik's cube with a few moves than it is to solve a scrambled Rubik's cube with a few moves. *[Optional -- Have the students discover this using a Rubik's cube or an online simulated cube].*

Make a connection to the previous lesson by comparing these to sorting algorithms, where some are speedy and efficient like Merge sort and Quick sort, and others are unusably slow, like Bubble sort. Highlight the difference that different problems have different lower bounds on optimal solutions, and that some problems like integer factorization have solutions but take too long to be solved in a practical way.

Computation [10 min]

Discuss the definition of computation (in a theoretical sense) with your students. Computation is input plus processing to get output. A computer is one system that is a "model of computation" since it takes input, processes it, and produces output.

Another model of computation is called a Turing machine, named after Alan Turing (one of the most famous computer scientists). A Turing machine is a theoretical entity that has a tape of symbols (a line of 0s and 1s), a head that can read only one symbol at a time, and an internal state that can change based on instructions as the head reads symbols. Turing and a mathematician called Alonzo Church are responsible for the "Church-Turing" thesis, which says that a Turing machine can compute anything that a digital computer can. This is a fundamental idea of the theory of computation, and has the implication that anything one computer is capable of doing is possible to be computed by another, given enough resources (time and memory).

Computability [15 min]

Now discuss the idea of computability with your students. Ask your students to answer or think-pair-share: are there things it is impossible for a computer to compute? The most classic "undecidable" (non-computable) question is called the Halting Problem. The Halting Problem is: make a program that can tell if another program will halt (terminate at some point eventually) or will loop forever and never end.

The Halting Problem is impossible for a computer to compute, which you can prove (informally) by paradox. Suppose you *did* have a program that solved the Halting Problem, called HALT(X), which takes the code for some program X as input and says "yes" if X terminates or "no" if X loops forever. Then you could write a new program that uses HALT inside it, which we will call PARADOX(X). First PARADOX(X) will run HALT(X) and if the result is no, PARADOX will halt, but if the result is yes, then PARADOX will loop forever. But here is the problem: what if we use the code for PARADOX as the input to PARADOX, running PARADOX(PARADOX)? If it says that PARADOX halts, then PARADOX runs forever, and if it says PARADOX runs forever, then PARADOX halts. This problem *is* a *paradox* and does not make sense because the premise, that a program called HALT could exist, must be wrong! Therefore, the Halting Problem is impossible for a computer to solve.

Video explanation with optional student simulation

- Alonzo Church, an American, and Alan Turing, from the UK, independently proved in the 1930s – before computers actually existed – that there are some problems that computers will never be able to solve. View: The Halting Problem at: https://(https://www.youtube.com/watch?v=92WHN-pAFCs)www.youtube.com/watch?v=92WHN-pAFCs (https://www.youtube.com/watch?v=92WHN-pAFCs) (https://www.youtube.com/watch?v=92WHN-pAFCs) [Optional 7:52] Have groups of students act out the machines in the video to determine whether they understand the basics of the proof.

Wrap Up (5 min)

Think-Pair-Share:

- Ask your students to think about the following algorithms, pair off, and reorder them based on worst-case computational complexity, with the fastest ones first and the slower (or undecidable) ones last:
  - Bubble Sort
  - Factoring large integers
  - Merge Sort
  - Binary Search
  - Taking attendance
  - The Halting Problem
- Discuss the orderings that a few groups came up with. Advanced groups could also try to guess the computational complexity:
  - Binary Search, O(log n)
  - Taking attendance, O(n) since you just read off the list in order
  - Merge Sort, O(n log n)
  - Bubble Sort, $O(n^2)$
  - Factoring large integers, $O(e^n)$, approximately exponential depending on the algorithm
  - The Halting Problem, undecidable

Session 2

Getting Started (5 min)

This session concerns advanced algorithms, in particular heuristic search, which is commonly used in artificial intelligence. Refresh your students' minds on the definitions of computation, computability, and undecidable problems. Additionally, mention the properties we consider when we compare algorithms:

- correctness
- ease of understanding
- elegance and style
- time/space efficiency

Guided Activity (45 min)

Search and Game Trees [15 min]

Introduce the idea of heuristic search, which is a class of algorithms used in many artificial intelligence programs. A heuristic is something that is used to find a good solution in a reasonable time, and a heuristic search algorithm is an algorithm that uses heuristics to determine how to search through some space.

A great way to introduce heuristic search is first to discuss game trees. A game tree is a structure that is used to represent the "space" of a game that an algorithm wants to search through.

Think of a game like chess: you make a move, the opponent makes a move, and the process continues until the ending conditions have been met (one player in checkmate or stalemate). A game tree is a mathematical structure used by AI and heuristic search algorithms to model the moves made in a chess game. At any turn, we can make a "tree" by drawing the root node as representing the current state of the board and drawing one branch under it for every possible move. In Tic-Tac-Toe, if you are the starting player, then the root node represents a blank board, and there will be nine branches, one for each possible move (each space where you could place your mark). Following a branch in the game tree takes you to a new node that represents the configuration of the game that results from having taken that move. In Tic-Tac-Toe, if I am the first player and place my X in the center space, I have "followed" that branch down the tree to a new node that represents the board with an X in the center space. The opponent then uses this node as the root of their game tree, and has a branch for each of their possible moves.

Think-Pair-Share: Have your students pair off and play a game of Tic-Tac-Toe and try to draw the game tree as they play it, drawing the nodes for each move they made and every potential branch from those nodes. Bring them back into discussion and ask them what if they had to draw out *every* node followed down *every* branch? Now ask them to imagine the game tree for chess, which has 20 possible moves on the first turn, 400 on the second, and many, many more as the game goes on. How can an artificially intelligent program learn to play chess when there are so many (too many) options? Chess actually has around 35^{100} nodes in its tree and 10^{40} legal states.

Game-Playing AI [10 min]

Heuristic search on game trees is one way AI programs are able to play games like chess. How good are computer game players?

- **Chess**: Deep Blue, a complex machine that used databases and heuristic search, beat Garry Kasparov (one of the best chess players ever) in 1997
  - Garry Kasparov vs. Deep Junior (Feb 2003): tie!
  - Kasparov vs. X3D Fritz (November 2003): tie!
  - http://www.cnn.com/2003/TECH/fun.games/11/19/kasparov.chess.ap/ (http://www.cnn.com/2003/TECH/fun.games/11/19/kasparov.chess.ap/)
- **Checkers**: Chinook is an AI program with a *very large* endgame database that is the world champion. Checkers, like Tic-Tac-Toe, is "solved," meaning there is a known optimal way to play the game to always win or force a tie. You can play a version of Chinook here:
  - https://webdocs.cs.ualberta.ca/~chinook/ (https://webdocs.cs.ualberta.ca/~chinook/)
- **Bridge**: "Expert-level" computer players exist (but no world champions yet).
- **Poker**: Computer team beat a human team, using statistical modeling and adaptation:
  - http://www.cs.ualberta.ca/~games/poker/man-machine/ (http://www.cs.ualberta.ca/~games/poker/man-machine/)

- Good places to learn more:
  - http://www.cs.ualberta.ca/~games/ (http://www.cs.ualberta.ca/~games/)
  - http://www.cs.unimass.nl/icga (http://www.cs.unimass.nl/icga)

Typically games modelled with game trees are 2-person games, players alternate moves, and they are zero-sum (meaning one player's loss is the other's gain). More complicated elements in such games may have include: hidden information (like other players' hands), chance (dice), or multiple players.

Playing a Game with Heuristic Search [10 min]

How does an AI program use heuristic search to play a game? Typically in these steps:

- consider all moves that are possible for the current turn
- compute what the new positions and configuration of the board (the "state") for each of those moves
- evaluate each state using some scoring function to determine which is better
  - for example, taking an opponent's piece is probably going to be evaluated more highly than simply moving a pawn
- make the move that results in the best evaluated state
- wait for your opponent to play, repeat

The key problems are:

- representing the state of the board
- generating the resulting states from every move
- evaluating the value of the resulting states

For evaluation, some function is typically coded or learned over time.

- For Tic-Tac-Toe, for board state n, an evaluation function could be:
  - f(n) = [# of 3-lengths open for me] - [# of 3-lengths open for you], where a 3-length is a complete row, column, or diagonal
- Alan Turing's function for chess with board state n:
  - f(n) = w(n)/b(n) where w(n) is the sum of the point value of white's pieces on the board at state n, and b(n) is the sum of black's pieces. The point values are those commonly used by professional chess players, pawn is 1 point, the queen is 9 points.

Types of Heuristic Search [10 min]

Refer to the "Advanced Algorithms" slides in the lesson resources folder for examples of uninformed search. For an activity, you may want to create a game tree for Tic-Tac-Toe and have your students walk through how each of the following algorithms would operate over it.

Uninformed Search are algorithms that work without a heuristic, using no information about the likely "direction" of the goal node. Algorithms include:

- depth-first search
  - starting at the root of the game tree, pick one branch and examine the node at the end of it, then pick one branch of that node and examine the even deeper node (hence, "depth-first") until you reach the end of the game, then go back one node, pick one of its branches, explore until you reach the end of the game, and so on
  - this search is unpractical for games with many moves that may go on indefinitely
- breadth-first search
  - starting at the root of the game tree, called A. It has n branches leading to child nodes each called B_1, B_2, and on to B_n, for some n nodes. Examine each of these children in order before moving onto the children of B_1, examining each of those in order, then examining the child nodes of B_2 then those of B_3, and so on.
  - this search is unpractical for games with many or variable numbers of moves per turn

For any games with variety and complexity, certainly for chess and even checkers, uninformed search is simply too slow because it is exhaustive. This problem is another example, like with sorting, where the efficiency of our solution matters a great deal. To get programs to play games, we need them to be efficient and intelligent about the number and quality of moves they consider.

Informed Search algorithms each follow some heuristic that uses information about the game to determine smart directions to explore. Examples include:

- best-first search ("greedy")
  - at every turn, take the branch leading to the node with the greatest value from the evaluation function
  - the problem here is when there is delayed reward since this "greedy" approach lacks any ability to look ahead. For instance, if there are two moves available, one that is great and another that is just decent, it will always pick the great move even if a winning move could be made the turn after the decent one.
- A* ("a star")
  - estimates the goal node and picks nodes based on the least cost. It follows a best-first strategy but also factors in the distance it has traveled from the original root node.
- Your GPS device! In order to figure out the best, fastest route to your destination, your GPS will search through the possible roads you can take intelligently by factoring in things such as the span and capacity of the road, the traffic, and potentially even the time of day.

Advanced classes may wish to discuss local search algorithms, such as hill-climbing and genetic algorithms (in the "Advanced Algorithms" slides in the lesson folder).

Minimax

Thinking about game trees again, we want to select the branch that takes us to a node with the maximum evaluated state. But there is a catch: the opponent gets to make moves, too. That is, every other branch in our game tree is the opponent's turn. How does the AI program account for the other player?

Perhaps most logically, the way AI programs do so is to assume the other player will play optimally. Just as the AI will take the branch that leads to the state with the greatest evaluation, it assumes the other player takes the branch leading to the state that will maximize *their* position. In other words, the AI searches through their game tree by following the branch with the maximum value on their turn, and following the branch with the minimum value on the opponent's turn. This algorithm is called minimax and is the basis of nearly all AI that play 2-person zero-sum games.

Options for Differentiated Instruction

For the Halting Problem proof, it is important that students can translate the solution that is on the video into a representation that makes (some) sense to them. Acting out the inputs and outputs of the set of machines is an approach worth trying.

Evidence of Learning

Formative Assessment

The following "Checks for Understanding" could be used to guide the students towards the three learning objectives.

Objective: Students will identify some Advanced Algorithms that Exploit Inverse Operations Efficiency.

1. Pairs of students will be asked to list pairs of basic arithmetic inverse functions.
2. The class will develop a composite list of inverse functions found by the student pairs. Note: many of these pairs share the same key on their graphing calculators.
3. Students will factor composite numbers and create the same composite numbers from their prime factorization. They will log the relative effort in their journals.

Objective: Students will identify some Advanced Algorithmic Techniques.

1. Students will find examples from earlier modules where the algorithms used techniques of heuristics, randomness, probability, etc. This could be a good group review of prior topics.

Objective: SWBAT discuss at least one example of a computing problems that is unsolvable

1. Students will either describe the proof of Turing's Halting problem using models in a way that is similar to that used in the lesson video or they wil build a similar physical model using students as the machines.

Summative Assessment

Students will be able to summarize -- in their own words or with simple models -- the proof of the Halting Problem.

Students will be able to identify the sensitivity of cryptography to the difficulty of factoring large numbers.

Internal Use Only

Pending Tasks

Structured like the theory/complexity slides from COMP 101 - covers solvability and complexity (leave out FSAs)

Add a lesson after this one that uses AI heuristic search and game playing as examples of advanced algorithms

(http://csmatters.org) 5 - 6

0b101 - 0b110

Create Performance Task Partial Practice

Unit 5. Data Manipulation

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 3 50-minute sessions

Lesson Summary

Summary

Students will define, design and implement a programming project: a miniature version of the Create Performance Task. They will create presentations and share with groups the projects they developed and how their project used abstractions.

Outcomes

- Students will complete a miniature version of the Create Performance Task.

Overview

Session 1:

1. Presentation (5 min) Students are introduced to the Create Task.
2. Activity (15 min) Students select a small programming project to model the Create Task.
3. Activity (10 min) Students identify an algorithm to use in their program and share within tabe groups.
4. Activity (10 min) Students identify an an abstraction to use in their program.
5. Wrap Up (5 min) Students share the algorithms and abstractions they will use with the class.

Session 2:

1. Journal (2 min) - What is your plan for today for the development of your project? What abstractions do you plan on using in your project?
2. Activity (43 min) - Students complete implementing their projects.
3. Journal (5 min) - Reflection. What abstractions did you use in your project?

Session 3:

1. Getting Started (5 min) - Students individually respond to three prompts about their projects.
2. Activity (20 min) - Students prepare one minute presentations of their projects.
3. Presentations (15 min) - Students present their project to table groups.
4. Wrap Up (5 min) - Students create exit slips with any questions about the Create Task.

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
- 1.2.1 - Create a computational artifact for creative expression.
  - 1.2.1B - Creating computational artifacts requires understanding of and use of software tools and services.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4D - Effective collaboration strategies enhance performance.
  - 1.2.4E - Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.
- 2.2.1 - Develop an abstraction when writing a program or creating other computational artifacts.
- 2.2.2 - Use multiple levels of abstraction to write programs.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
  - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
  - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
  - 4.1.1E - Algorithms can be combined to make new algorithms.

    - 4.1.1F - Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.
    - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
    - 4.1.1H - Different algorithms can be developed to solve the same problem.
    - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
    - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
    - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
    - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
    - 4.1.2D - Different languages are better suited for expressing different algorithms.
    - 4.1.2F - The language used to express an algorithm can affect characteristics such as clarity or readability but not whether an algorithmic solution exists.
    - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
    - 4.1.2H - Nearly all programming languages are equivalent in terms of being able to express any algorithm.
    - 4.1.2I - Clarity and readability are important considerations when expressing an algorithm in a language.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
    - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
    - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
    - 5.1.1C - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution.
    - 5.1.1D - Additional desired outcomes may be realized independently of the original purpose of the program.
    - 5.1.1E - A computer program or the results of running a program may be rapidly shared with a large number of users and can have widespread impact on individuals, organizations, and society.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
    - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
    - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
    - 5.1.2D - Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.
    - 5.1.2E - Documentation about program components, such as code segments and procedures, helps in developing and maintaining programs.
    - 5.1.2F - Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments.
    - 5.1.2G - Program development includes identifying programmer and user concerns that affect the solution to problems.
    - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.
    - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.1.3 - Collaborate to develop a program.
    - 5.1.3A - Collaboration can decrease the size and complexity of tasks required of individual programmers.
    - 5.1.3B - Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.
    - 5.1.3C - Collaboration in the iterative development of a program requires different skills than developing a program alone.
    - 5.1.3D - Collaboration can make it easier to find and correct errors when developing programs.
    - 5.1.3E - Collaboration facilitates developing program components independently.
    - 5.1.3F - Effective communication between participants is required for successful collaboration when developing programs.
- 5.2.1 - Explain how programs implement algorithms.
    - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
    - 5.2.1B - Program instructions are executed sequentially.
    - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
    - 5.2.1E - Program execution automates processes.
    - 5.2.1G - A process may execute by itself or with other processes.
    - 5.2.1J - Simple algorithms can solve a large set of problems when automated.
- 5.3.1 - Use abstraction to manage complexity in programs.
    - 5.3.1A - Procedures are reusable programming abstractions.
    - 5.3.1B - A procedure is a named grouping of programming instructions.
    - 5.3.1C - Procedures reduce the complexity of writing and maintaining programs.
    - 5.3.1D - Procedures have names and may have parameters and return values.
    - 5.3.1E - Parameterization can generalize a specific solution.
    - 5.3.1F - Parameters generalize a solution by allowing a procedure to be used instead of duplicated code.
    - 5.3.1G - Parameters provide different values as input to procedures when they are called in a program.
    - 5.3.1I - Strings and string operations, including concatenation and some form of substring, are common in many programs.
    - 5.3.1J - Integers and floating-point numbers are used in programs without requiring understanding of how they are implemented.
    - 5.3.1L - Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.
    - 5.3.1M - Application program interfaces (APIs) and libraries simplify complex programming tasks.
    - 5.3.1N - Documentation for an API/library is an important aspect of programming.
    - 5.3.1O - APIs connect software components, allowing them to communicate.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1A - Program style can affect the determination of program correctness.

- 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
- 5.4.1D - Longer code segments are harder to reason about than shorter code segments in a program.
- 5.4.1E - Locating and correcting errors in a program is called debugging the program.
- 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.
- 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
- 5.4.1I - Programmers justify and explain a program's correctness.
- 5.4.1J - Justification can include a written explanation about how a program meets its specifications.
- 5.4.1K - Correctness of a program depends on correctness of program components, including code segments and procedures.
- 5.4.1L - An explanation of a program helps people understand the functionality and purpose of it.

- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
  - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
  - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
  - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
  - 5.5.1F - Compound expressions using and, or, and not are part of most programming languages.
  - 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.
  - 5.5.1H - Computational methods may use lists and collections to solve problems.
  - 5.5.1I - Lists and other collections can be treated as abstract data types (ADTs) in developing programs.
  - 5.5.1J - Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection.

Key Concepts

Students practice choosing a project and planning how to implement it in a fixed time frame.

Students have just two days to plan and implement a project. Since these will be small projects, students may need help using algorithms and data abstraction. Since an algorithm is a list of steps that comes to a conclusion, if students develop pseudocode for their projects they can refer to the pseudocode as their algorithm.

Students may receive most of the credit from an incomplete project if the project demonstrates the required components.

For this practice task, teachers may want to provide program stubs. Stubs could include suggested functions.

Essential Questions

- How are algorithms implemented and executed on computers and computational devices?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?

Teacher Resources

Lesson Plan

Session 1: Planning Day

Present an overview of the Create Task.

Explain that students will have 12 hours to complete the Create Task later in the course and they will three 50-minute sessions for this practice. The actual Create Task will have a suggested collaborative component and be larger in scope.

Discuss the following guidelines for the full project and the practice project we will be doing.

Full Create Task Guidelines

Three components to create:

- Program
- Report
- Video

General:

One project - individual with collaboration in stages

12 hours of classroom time

Project must use functional and data abstraction.

Report: Written responses (response to all prompts combined must not exceed 750 words, exclusive of the Program Code.):

a. Provide a written response or audio narration in your video that: Identifies the programming language and identifies the purpose of your program.

b. Describe the incremental and iterative development process of your program

c. Describe how a selected algorithm functions.

d. Explain how an abstraction you developed helped manage complexity

Practice Create Task Guidelines

For this practice task, students will complete simpler project and a one-minute presentation about it, rather than a video and a report.

Students work individually to select projects.

After completing the project, students will create a one-minute presentation about it. Presentation should address the following.

a. Identify the programming language and the purpose of your program.

b. Describe the incremental and iterative development process of your program

c. Describe how a selected algorithm functions.

d. Explain how an abstraction you developed helped manage complexity

The presentation must address at least points a and b of the above and c or d.

Projects are chosen by the student. If they wish, their projects may be based on the following labs from *How to Think Like a Computer Scientist*.

Labs

- Astronomy Animation (http://interactivepython.org/runestone/static/thinkcspy/Labs/astronomylab.html)
- Turtle Racing Lab (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab03_01.html)
- Drawing a Circle (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01.html)
- Lessons from a Triangle (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01a.html)
- Finally a Circle (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab04_01a.html#finally-a-circle)
- Counting Letters (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab12_01.html)
- Letter Count Histogram (http://interactivepython.org/runestone/static/thinkcspy/Labs/lab12_02.html)
- Approximating the Value of Pi (http://interactivepython.org/runestone/static/thinkcspy/Labs/montepi.html)
- Python Beyond the Browser (http://interactivepython.org/runestone/static/thinkcspy/Labs/pythonshell.html)
- Experimenting With the 3n+1 Sequence (http://interactivepython.org/runestone/static/thinkcspy/Labs/sequencelab.html)
- Plotting a sine Wave (http://interactivepython.org/runestone/static/thinkcspy/Labs/sinlab.html)

Students select a project and share their ideas with partners.

After collaborating with partners, students submit to their teacher a brief description of the project describing its most important features and how it will work.

Students identify an algorithm and an abstraction to use in their projects. The algorithm should be written as pseudocode and then shared with their partners.

Students should discuss in groups what abstraction they chose and how they think it would be helpful.

Session 2: Implementation Day

Warm up (2 min)

Students complete a brief journal entry describing:

- Their plan for today in the development of the project.
- What abstractions they will be using in the project.

Work Time (43 min)

Students work to implement and test projects. Teachers may evaluate student performance based on student journal entries and their observations of their effort in implementing the project.

Closing (5 min)

Students reflect on their project and making journal entry of how they used abstraction in the project.

Day 3: Presentation Day

Warmup (5 min)

Students begin by individually responding to these prompts about their project:

b. describe the purpose, how your program code works and the most important features and algorithms

d. describe the development process

e. explain an abstraction and how it helped manage complexity

Presentation Preparation (20 min)

Students prepare one-minute presentation about their projects including their responses to prompts b, d and e.

Presentations (15 min)

Students present their project to table groups. Time the presentations so that they do not exceed 1 minute. Students share with table groups what they like about the project, what they learned and any questions they have.

Closing (5 min)

Students create exit slips with any questions they have about the Create Task after viewing and discussing the presentations.

Evidence of Learning

Formative Assessment

For the practice task, project descriptions and pseudocode for each proposed project should be assessed. Assessment can be done by collaborative partners first. If partners have concerns, they should be brought to the teacher. If student projects are too big or too small in scope, teachers should provide feedback.

Summative Assessment

The project should be scored using the latest rubric provided by the College Board.

The latest rubric (updated as of June 2016) is in the lesson folder.

Revision Date: Aug 19, 2016 (Version 2.0)
Duration: 5 50-minute sessions

Lesson Summary

Summary

EarSketch teaches computer science through music composition and remixing. No prior knowledge of either computer science or music is needed. Students can express their own unique style. EarSketch also lends itself well to student collaboration as well as a discussion on proprietary ownership.

EarSketch consists of three components:

1. A free online curriculum that teaches programming concepts using Python while teaching music composition and remixing.
2. A free online software toolset, which contains a code editor to write and test Python code and a Digital Audio Workstation (DAW) to actually play the music.
3. A social media website. **EarSketch is retiring the social media site before the 2015-2016 school year.** Students may collaborate via the social media website this year if teachers desire. Under the terms of the social media site's use, shared files are donated to the community. Students currently create an account on the site to gain access to cloud file storage for their files. **No other use of the file sharing site is required either by EarSketch or CS Matters lessons.**

Students create an account to get Cloud storage for their files.

- This curriculum does not ask students to post anything on this or on any public web site.
- The curriculum does provide a place to review advantages and pitfalls of social media sites. Especially note that student projects posted on this site are no longer the property of the student artist and are freely open for anyone to use as their own.
- No downloads or installs are needed. EarSketch runs inside a web browser with recent versions of Chrome, Firefox, or Safari. (Internet Explorer < 12 is not supported.)
- **Students need ear buds or headphones for these lessons.**

Outcomes

- Students will understand the basics of music including beat, measure, track, and effects.
- Students will use the Python programming language to create and remix their own music.
- Students will apply Python programming concepts - iteration, user-defined functions, debugging

Optionally from Sections 2 and 3.

- Students will apply Python programming concepts - list creation, access, modification and traversal
- Students will use the Python programming language to create and remix their own music including effects and musical forms
- Students will use the Python programming language to create and remix their own music including randomness and stochastic composition
- Students will explore sonification -a way to use non-speech audio to convey information, or in other words, turning data into sound.
- Students use Python to enable the computer to analyze audio.
- Students will implement recursive Python programs

Overview

The Lesson is divided into three sections.

Section 1 is anticipated to take about 5 sessions to complete these EarSketch units.

- 1 EarSketch Welcome and Unit 1 (Getting Started) (http://earsketch.gatech.edu/uncategorized/unit-1) [1 session]
- 2 EarSketch Unit 2: Effects and Beats (http://earsketch.gatech.edu/category/unit-2) [1 session]
- 3 EarSketch Unit 3: For Loops, User-Defined Functions, Debugging (http://earsketch.gatech.edu/category/unit-3) [1 session]
- 4 Project 1 [1 or two sessions]
- 5 Summative Assessment 1

Section 2 is anticipated to take about 5 sessions to complete these EarSketch units.

- 6 EarSketch Unit 4: Lists, Effects in More Detail, Musical Form (http://earsketch.gatech.edu/category/unit-4) [1 session]
- 7 EarSketch API [.5 session]
- 7 EarSketch Unit 5: Randomness (http://earsketch.gatech.edu/category/unit-5) [.5 session]
- 8 EarSketch Unit 6: Sonification (http://earsketch.gatech.edu/category/unit-6) [1 session]
- 9 Project 2 [1 or 2 sessions]
- 10 Summative Assessment 2

Section 3 is **optional** and is anticipated to also take about 5 sessions to complete these Earsketch units.

- 11 EarSketch Unit 7: Teaching Computers to Listen (http://earsketch.gatech.edu/category/unit-7) [1 session]
- 12 EarSketch Unit 9: Recursion (http://earsketch.gatech.edu/category/unit-9) [1 session]
- 13 Project 3 [2 sessions]
- 14 Summative Assessment

Each session will have the following elements.

1. Getting Started: (5 min)

2. Guided Activities (5 sessions)
3. Wrap Up (5 min)

Sources

The Earsketch lessons are all taken from http://earsketch.gatech.edu/category/learning/welcome (http://earsketch.gatech.edu/category/learning/welcome).

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.1 - Create a computational artifact for creative expression.
  - 1.2.1C - Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming integrated development environments (IDEs), spreadsheets, three-dimensional (3-D) printers, or text editors.
  - 1.2.1E - Creative expressions in a computational artifact can reflect personal expressions of ideas or interests.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
  - 1.2.3B - Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision.
  - 1.2.3C - Combining or modifying existing artifacts can show personal expression of ideas.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4B - Effective collaborative teams consider the use of online collaborative tools.
  - 1.2.4C - Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation.
  - 1.2.4E - Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.
  - 1.2.4F - A collaboratively created computational artifact can reflect personal expressions of ideas.
- 1.2.5 - Analyze the correctness, usability, functionality, and suitability of computational artifacts.
  - 1.2.5A - The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact.
  - 1.2.5D - The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.
- 1.3.1 - Use computing tools and techniques for creative expression.
- 2.1.1 - Describe the variety of abstractions used to represent data.
  - 2.1.1A - Digital data is represented by abstractions at different levels.
  - 2.1.1C - At a higher level, bits are grouped to represent abstractions, including but not limited to numbers, characters, and color.
- 2.2.1 - Develop an abstraction when writing a program or creating other computational artifacts.
  - 2.2.1A - The process of developing an abstraction involves removing detail and generalizing functionality.
  - 2.2.1C - An abstraction generalizes functionality with input parameters that allow software reuse.
- 2.2.2 - Use multiple levels of abstraction to write programs.
  - 2.2.2B - Being aware of and using multiple levels of abstractions in developing programs help to more effectively apply available resources and tools to solve problems.
- 2.2.3 - Identify multiple levels of abstractions used when writing programs.
  - 2.2.3K - Lower-level abstractions can be combined to make higher-level abstractions, such as short message services (SMS) or email messages, images, audio files, and videos.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
  - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
  - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
  - 4.1.1E - Algorithms can be combined to make new algorithms.
  - 4.1.1F - Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.
- 4.1.2 - Express an algorithm in a language.
  - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
  - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
  - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
  - 4.1.2E - Some programming languages are designed for specific domains and are better for expressing algorithms in those domains.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
  - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
  - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.

    - 5.1.1C - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution.
    - 5.1.1E - A computer program or the results of running a program may be rapidly shared with a large number of users and can have widespread impact on individuals, organizations, and society.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
    - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
    - 5.1.2D - Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.
    - 5.1.2E - Documentation about program components, such as code segments and procedures, helps in developing and maintaining programs.
    - 5.1.2F - Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments.
    - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.
    - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.1.3 - Collaborate to develop a program.
    - 5.1.3A - Collaboration can decrease the size and complexity of tasks required of individual programmers.
    - 5.1.3B - Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.
    - 5.1.3C - Collaboration in the iterative development of a program requires different skills than developing a program alone.
    - 5.1.3D - Collaboration can make it easier to find and correct errors when developing programs.
    - 5.1.3E - Collaboration facilitates developing program components independently.
    - 5.1.3F - Effective communication between participants is required for successful collaboration when developing programs.
- 5.2.1 - Explain how programs implement algorithms.
    - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
    - 5.2.1B - Program instructions are executed sequentially.
    - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
- 5.3.1 - Use abstraction to manage complexity in programs.
    - 5.3.1A - Procedures are reusable programming abstractions.
    - 5.3.1B - A procedure is a named grouping of programming instructions.
    - 5.3.1C - Procedures reduce the complexity of writing and maintaining programs.
    - 5.3.1D - Procedures have names and may have parameters and return values.
    - 5.3.1E - Parameterization can generalize a specific solution.
    - 5.3.1F - Parameters generalize a solution by allowing a procedure to be used instead of duplicated code.
    - 5.3.1G - Parameters provide different values as input to procedures when they are called in a program.
    - 5.3.1H - Data abstraction provides a means of separating behavior from implementation.
    - 5.3.1L - Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.
    - 5.3.1M - Application program interfaces (APIs) and libraries simplify complex programming tasks.
    - 5.3.1N - Documentation for an API/library is an important aspect of programming.
    - 5.3.1O - APIs connect software components, allowing them to communicate.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1A - Program style can affect the determination of program correctness.
    - 5.4.1B - Duplicated code can make it harder to reason about a program.
    - 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
    - 5.4.1D - Longer code segments are harder to reason about than shorter code segments in a program.
    - 5.4.1E - Locating and correcting errors in a program is called debugging the program.
    - 5.4.1F - Knowledge of what a program is supposed to do is required in order to find most program errors.
    - 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
    - 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
    - 5.4.1I - Programmers justify and explain a program’s correctness.
    - 5.4.1J - Justification can include a written explanation about how a program meets its specifications.
    - 5.4.1K - Correctness of a program depends on correctness of program components, including code segments and procedures.
- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
    - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
    - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
    - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
    - 5.5.1F - Compound expressions using and, or, and not are part of most programming languages.
    - 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.
- 7.2.1 - Explain how computing has impacted innovations in other fields.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.

2.1.1 AC

2.2.1 AC

2.2.2 B

2.2.3 K

5.1.1 B

5.1.3 ABCDEF

5.3.1 ABCDEFGHIKL

5.4.1 CEK

5.5.1 AHI

7.2.1 BC

7.3.1 ABC

Math Common Core Practice:

- MP1: Make sense of problems and persevere in solving them.
- MP2: Reason abstractly and quantitatively.
- MP5: Use appropriate tools strategically.
- MP6: Attend to precision.
- MP7: Look for and make use of structure.
- MP8: Look for and express regularity in repeated reasoning.

Common Core Math:

- F-IF.1-3: Understand the concept of a function and use function notation
- F-BF.1-2: Build a function that models a relationship between two quantities
- F-LE.5: Interpret expressions for functions in terms of the situation they model

Common Core ELA:

- RST 12.3 - Precisely follow a complex multistep procedure
- RST 12.4 - Determine the meaning of symbols, key terms, and other domain-specific words and phrases
- RST 12.7 - Integrate and evaluate multiple sources of information presented in diverse formats and media
- RST 12.9 - Synthesize information from a range of sources
- WHST 12.2 - Write informative/explanatory texts, including the narration of historical events, scientific procedures/experiments, or technical processes
- WHST 12.4 - Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience
- WHST 12.6 - Use technology, including the Internet, to produce, publish, and update writing products

NGSS Practices:

- 2. Developing and using models
- 3. Planning and carrying out investigations
- 4. Analyzing and interpreting data
- 5. Using mathematics and computational thinking

NGSS Content:

- HS-ETS1-2. Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.

Essential Questions

- How can a creative development process affect the creation of computational artifacts?
- How can computing and the use of computational tools foster creative expression?
- How can computing extend traditional forms of human expression and experience?
- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How does abstraction make the development of computer programs possible?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?
- How does computing enable innovation?
- What are some potential beneficial and harmful effects of computing?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

Teacher Resources

Student computer usage for this lesson is: **required**

Students will need earbuds or headphones for these lessons.

EarSketch consists of three components:

- an integrated curriculum: http://earsketch.gatech.edu/ (http://earsketch.gatech.edu/)
- a software toolset: http://earsketch.gatech.edu/earsketch2/ (http://earsketch.gatech.edu/earsketch2/)
- a social media website: http://earsketch.gatech.edu/media/ (http://earsketch.gatech.edu/media/) discontinued after 2015

The software toolset component includes the EarSketch code editor and digital audio workstation environment to write code and make music. It runs inside a web browser with the latest versions of Chrome, FireFox, or Safari. **Internet Explorer** is **not** supported and the digital audio workstation will not load. You must use a browser that supports Web Audio. (Internet Explorer 12 plans to include support for Web Audio.)

Teachers should review the first two modules of the student curriculum to learn the components of EarSketch: Unit 1 (Getting Started) (http://earsketch.gatech.edu/uncategorized/unit-1) and Unit 2 (Effects and Beats) (http://earsketch.gatech.edu/uncategorized/unit-2).

Next, teachers should access the teacher curriculum (http://earsketch.gatech.edu/teaching-resources-reference/teacher-curriculum-module-1), which is designed to help computer science teachers **with little or no music knowledge** begin teaching EarSketch in their classrooms. It presents music concepts, rhythms, pattern and variety, and effects as they relate to music programming in EarSketch.

Finally, teachers should complete the student curriculum to get an idea of what students will be learning and doing.

Lesson Plan

Section 1 Introduction to EarSketch

Session 1

Getting Started (5 min)

Journal Prompt: What are possible advantages there are to creating and mixing music on a computer?

Responses should be collected from each student and used to create word cloud. Project the following four benefits to programming music and ask, "Are any of these missing?"

1. You can automate repetitive, tedious tasks.
2. You can experiment with music more easily.
3. You can roll die. (Introducing randomization into music.)
4. You can turn data into music. (What happens if you interpret data in a musical way?)

Students should select one of these four points and record throughts and observations as to their meanings.

Guided Activities (40 min)

Direct students to Unit 1: Getting Started with EarSketch (http://earsketch.gatech.edu/uncategorized/unit-1) (http://earsketch.gatech.edu/uncategorized/unit-1 (http://earsketch.gatech.edu/uncategorized/unit-1)).

Within the first unit, they should explore the section 1.1. Introduction to the DAW (Digital Audio Workstations) (http://earsketch.gatech.edu/uncategorized/unit-1#chap11) (http://earsketch.gatech.edu/uncategorized/unit-1#chap11 (http://earsketch.gatech.edu/uncategorized/unit-1#chap11)). Students should research the following definitions and procedures, then share them with a partner.

Definitions

1. **DAW**
2. **measure**
3. **track**

Procedures

1. How does one play (run) an EarSketch script?
2. How does one isolate and play just one track in an EarSketch script?

Once students have completed this task, demonstrate the following sections for them:

1. 1.3. Anatomy of an EarSketch Project (http://earsketch.gatech.edu/uncategorized/unit-1#chap131) (http://earsketch.gatech.edu/uncategorized/unit-1#chap131 (http://earsketch.gatech.edu/uncategorized/unit-1#chap131)). Especially important is the process of creating, opening, running, editing, and saving EarSketch python scripts both on the EarSketch cloud as well as in the classroom.
2. 1.4. Intro to Music Programming in EarSketch (http://earsketch.gatech.edu/uncategorized/unit-1#chap140) (http://earsketch.gatech.edu/uncategorized/unit-1#chap140 (http://earsketch.gatech.edu/uncategorized/unit-1#chap140)). Especially important is the structure of an EarSketch script, the use of constants and variables, and the purpose and usage of fitMedia().

It is important that students know how to use the curriculum and the online development environment as the subsequent sessions will have students working independently with these tools.

Wrap Up (5 min)

Students should reflect on the questions and thoughts they recorded at the beginning of class. They should reflect on the following:

1. Which questions have been answered?
2. Which questions remain?
3. What new questions arose?

Assignment

Assignment 1.1

This assignment can be found within the curriculum resources at Unit 6 > Lesson 1 > EarSketch Units > Unit 1.

Session 2

Getting Started (5 min)

Journal: Students should open their Assigment 1.1 (homework from the previous section) and discuss with their elbow partners what they learned while completing it. They should record in their journals two insightful observations made either in the previous session or during the completion of the assignment. Any questions remaining after these discussions should be posted up by students.

Guided Activities (40 min)

Direct students to Unit 2: Effects and Beats (http://earsketch.gatech.edu/uncategorized/unit-2) (http://earsketch.gatech.edu/uncategorized/unit-2 (http://earsketch.gatech.edu/uncategorized/unit-2)).

Beats & Effects (25 min)

Within the second unit, they should go through each of the following sections and complete the exercises, making the appropriate modifications to the EarSketch script introduced and modified in Unit 1:

1. 2.1 Intro to Effects (http://earsketch.gatech.edu/uncategorized/unit-2#chap21) (http://earsketch.gatech.edu/uncategorized/unit-2#chap21 (http://earsketch.gatech.edu/uncategorized/unit-2#chap21)).
2. 2.2 Intro to Beats (http://earsketch.gatech.edu/uncategorized/unit-2#chap22) (http://earsketch.gatech.edu/uncategorized/unit-2#chap22 (http://earsketch.gatech.edu/uncategorized/unit-2#chap22)).

Once these exercises are completed, students should provide written responses to the following questions:

1. What is an effect?
2. How does one use setEffect()?
3. TASK: Describe beats, meter, and rhythm.
4. How can strings be used to create custom beat patterns?

Pair Programming (15 min)

Students should work in pairs to complete Quiz 2.1.

Wrap Up (5 min)

Spend some time identifying and responding to issues students might have run into.

Assignment

Students with registered accounts on EarSketch can save and name files in the cloud which simplifies file managment. Distribute and assign Assignment 2.2.

Session 3

Getting Started (5 min)

Collect: Assignments 1.1 and 2.2

Journal: Students should discuss with their elbow partners the first two assignments, reflecting on the lessons learned and identifying any questions that are lingering. Unresolved questions should be posted on the board.

Guided Activities (40 min)

Some time should be used here to respond to questions students posted at the beginning of the session. If questions are regarding upcoming material, answer these during the next activity.

Unit 3 Exercises & Prompts (25 min)

Direct students to Unit 3: For Loops, User-Defined Functions, Debugging (http://earsketch.gatech.edu/category/unit-3) (http://earsketch.gatech.edu/category/unit-3 (http://earsketch.gatech.edu/category/unit-3)).

Students should individually complete the exercises presented at the end of the following sections:

1. 3.1 For Loops (http://earsketch.gatech.edu/category/unit-3#chap41) (http://earsketch.gatech.edu/category/unit-3#chap41 (http://earsketch.gatech.edu/category/unit-3#chap41)).
2. 3.4 Debugging (http://earsketch.gatech.edu/category/unit-3#chap33) (http://earsketch.gatech.edu/category/unit-3#chap33 (http://earsketch.gatech.edu/category/unit-3#chap33)).

The Python encountered in these sections should be review. Students should write responses (individually) to the following:

1. What values are assigned to the variable i by the following Python statement: ***for i in range(1,10)***
2. What is a fill?
3. What boolean or logical operators are available in Python?
4. How can a variable be made to simultaneously hold two values?
5. TASK: Describe two strategies for debugging programs.

Pair Programming (15 min)

Students should get into pairs and complete Quiz 3.1.

Wrap Up (5 min)

Collect the quiz. Students should share responses to the 5 prompts from earlier in the session. Remind students that they will review Assignment 3.1 at the start of the next class just as they did today for Assignment 2.2. In the next class, students will collaborate to create the Section 1 project based on EarSketch units 1 - 3.

Assignment

Students should complete Assignment 3.1 for the next session.

Session 4

Getting Started (5 min)

Journal: Students should discuss with elbow partners the lessons they learned from session 3 (including the assignment and quiz). Any unresolved questions should be posted to the board.

Guided Activities (40 min)

Before continuing, any questions regarding last session's formative quiz or assignment 3.1 should be answered.

Students should work together to complete the project for units 1 through 3. This project can be found under Unit 6 Resources > Lesson 01 - Earsketch > Projects > Project Unit 1-3_1.docx. The final product should be one working program per group.

Wrap Up (5 min)

Students should reflect on what they liked about cooperating with one another and on what they want to improve in preparation for the upcoming "Create" performance tasks.

Assignment

Students should prepare for the section 1 exam using the assignments, quizzes and EarSketch units 1-3 as resources.

Session 5 (Section 1 Assessment)

Getting Started (5 min)

Students should upload their collaborative projects from session 4, including the .wav output of the music they created.

Guided Activities (40 min)

Students will take the section 1 test (units 1-3). The test to be administered can be found at the following location: Unit 6 Resources > Lesson 01 - Earsketch > Section Tests > Unit 1-3 Test.docx.

Time permitting, discuss the current music sharing sight and the ethical issues surrounding public sites. If students require more time to complete their collaborative projects, some could be given here as well.

Wrap Up (5 min)

Going foward, host a version of March Musical Madness. Hold a single elimination tournament to select the class musical section 1 champion. If going on to sections two and three, consider doing just one round of the contest. Each week allow pairs to enter their best product either from something newly created or modified.

Section 2 Dynamic Music Generation (Optional)

Session 6

Getting Started (5 mins)

Students should brainstorm lessons learned from the first section. Since they will be working in pairs throughout this entire section (excluding the project and the exam), this is a good time to also discuss standards for collaboration and cooperation. Partners and work groups should be specified here.

Notes gathered during this section, along with the EarSketch API documentation, will be allowed for the section 2 exam. This means that students should take good notes and save them in order to increase their scores on the test.

Guided Activities (40 mins)

Unit 4 (25 mins)

Direct students to Unit 4: Lists, Effects in More Detail, Musical Form (http://earsketch.gatech.edu/category/unit-4) (http://earsketch.gatech.edu/category/unit-4 (http://earsketch.gatech.edu/category/unit-4)).

Working in pairs, they should tackle the sections in this unit.

Formative Assessment (15 mins)

Working in pairs, students should complete Quiz 4. This quiz can be found under resources at the following location: Unit 6 Resources > Lesson 01 - EarSketch > EarSketch Units > Unit 4.

Wrap Up (5 mins)

Students should identify questions or concerns and share those that they think are most important, supplementing their notes with important comments.

Assignment

Select two assignments from the following list (these assignments can be found under the following: Unit 6 Resources > Lesson 01 - EarSketch > EarSketch Units > Unit 4).

1. Assignment 4.1
2. Assignment 4.2
3. Assignment 4.3
4. Assignment 4.4
5. Assignment 4.5

Session 7

Getting Started (5 mins)

Assess student progress from the previous session. Student working groups should identify questions or concerns. It is crucial that major concerns are addressed as soon as possible.

Guided Activites (40 min)

EarSketch API & Unit 5 (25 min)

Introduce the students to the EarSketch API (http://earsketch.gatech.edu/category/learning/reference/earsketch-api) (http://earsketch.gatech.edu/category/learning/reference/earsketch-api (http://earsketch.gatech.edu/category/learning/reference/earsketch-api)). Important is to impart the usefulness of this document - how might somebody take advantage of an API to accomplish a certain task?

After the API has been properly introduced, direct students to Unit 5: Randomness (http://earsketch.gatech.edu/category/unit-5) (http://earsketch.gatech.edu/category/unit-5 (http://earsketch.gatech.edu/category/unit-5)). Working in their groups, students should work through this EarSketch unit.

Formative Assessment (15 min)

Working in pairs, students should complete Quiz 5.1. This quiz can be found under resources at the following location: Unit 6 Resources > Lesson 01 - EarSketch > EarSketch Units > Unit 5.

Warp Up (5 min)

Students should identify questions or concerns and share those that they think are most important, supplementing their notes with important comments.

Assignment

Select and assign one of the following to assignments (these assignments can be found under the following: Unit 6 Resources > Lesson 01 - EarSketch > EarSketch Units > Unit 5):

1. Assignment 5.1
2. Assignment 5.2

Session 8

Getting Started (5 mins)

Assess student progress from the previous session. Student working groups should identify questions or concerns, making sure to share all concerns pertaining covered material before the following session.

Guided Activities (40 mins)

Unit 5 (25 min)

Direct students to Unit 6: Sonification (http://earsketch.gatech.edu/category/unit-6) (http://earsketch.gatech.edu/category/unit-6 (http://earsketch.gatech.edu/category/unit-6)).

Working in pairs, they should tackle the sections in this unit.

Formative Assessment (15 min)

Working in pairs, students should complete Quiz 6.1. This quiz can be found under resources at the following location: Unit 6 Resources > Lesson 01 - EarSketch > EarSketch Units > Unit 6.

Warp Up (5 min)

Students should identify questions or concerns and share those that they think are most important, supplementing their notes with important comments.

Assignment

Assignment 6.1

This assignment can be found within the curriculum resources at Unit 6 Resources > Lesson 01 - EarSketch > EarSketch Units > Unit 6.

Session 9

Getting Started (5 mins)

Assess student progress from the previous session. Student working groups should identify questions or concerns, making sure to share all concerns pertaining covered material before the following session.

Guided Activites (45 mins)

Have your students select and complete (in pairs) one of the following two projects. These projects can be found under Unit 6 Resources > Lesson 01 - EarSketch > Projects:

1. Project Unit 4-6_1.docx
2. Project Unit 4-6_2.docx

Assignment

Students should consolidate their notes and prepare for the section 2 exam next session. REMINDER: It is an open note exam.

Session 10

Getting Started (5 mins)

Journal: Have students reflect on the ethical concerns raised by digital media innovations such as EarSketch. In their journals, they should write down a specific ethical issue associated with the use of EarSketch. They should also point out one additional piece of online technology that raises digital media related ethical concerns.

Guided Activity (40 mins)

Distribute and administer the section 2 exam. This exam can be found at the following location: Unit 6 Resources > Lesson 01 - EarSketch > Section Tests > Unit 4-6 Test.docx.

Wrap Up (5 mins)

Host a version of March Musical Madness. Hold a single elimination tournament to select the class musical section 2 champion. If going on to section three, consider doing just one round of the contest. Each week allow pairs to enter their best product. It can be derived from the in-class work or be a completely new, out-of-class creation.

Section 3 Teaching Computers to Listen (Optional)

Section 3 is optional and is anticipated to also take about 5 sessions to complete these EarSketch units. Students should be working very independently during this unit in prepartion for the Create Task.

11 EarSketch Unit 7: Teaching Computers (http://earsketch.gatech.edu/category/unit-7) [1 session]

12 EarSketch Unit 9: Recursion (http://earsketch.gatech.edu/category/unit-9) [1 session]

13 Project 3 [2 sessions]

14 Summative Assessment

Session 11:

Getting Started

Guided Activities

Unit 7: Teaching Computers (http://earsketch.gatech.edu/category/unit-7)

Formative Assessment

Quiz 7.1.docx (https://drive.google.com/file/d/0ByBRH8unk6_5bHVQLWdibHRjams/view?usp=sharing)

Assignment:

7.1 (https://drive.google.com/open?id=0ByBRH8unk6_5Wi1xR2hzZmN2ejQ&authuser=0)

Wrap Up:

In pairs. then study groups students identify questions or concerns and share those they think are most important and supplement their notes with important comments.

Session 12:

Getting Started

Assess student progress from Session 11. Groups identify questions or concerns and share any items of concern before day 2 of this section.

Guided Activities

Unit 9: Recursion (http://earsketch.gatech.edu/category/unit-9)

Formative Assessment

Quiz 9.1 (https://drive.google.com/open?id=0ByBRH8unk6_5eHp5SzVLWk5XcTA&authuser=0)

Assignment:

9.1 (https://drive.google.com/open?id=0ByBRH8unk6_5YU9WYjhTeXJsRlk&authuser=0)

Wrap Up:

In pairs. then study groups students identify questions or concerns and share those they think are most important and supplement their notes with important comments.

Session 13 and 14:

Getting Started

Assess student progress from Session 12. Groups identify questions or concerns and share any items of concern before days 3 and 4 of this section.

Guided Activities

Project Unit 7-9_1.docx (https://drive.google.com/file/d/0ByBRH8unk6_5NEczWkRwQjNncUk/view?usp=sharing)

Project Unit 7-9_2.docx (https://drive.google.com/file/d/0ByBRH8unk6_5NzBxX081bHFPZlE/view?usp=sharing)

Assignment:

Prepare for Section 3 exam.

Wrap Up:

In pairs. then study groups students identify questions or concerns and share those they think are most important and supplement their notes with important comments.

Session 15:

Guided Activities

Unit 7-9 Test.docx (https://drive.google.com/file/d/0ByBRH8unk6_5U0ZOUnhRcjdQNTA/view?usp=sharing)

Wrap Up:

Host a version of March Musical Madness. Hold a single elimination tournament to select the class musical Section 2 champion. If going on to three consider doing just one round of the contest. Each week allow pairs to enter their best product either from somethnnewly created or modified.

Assignment:

None

Evidence of Learning

Formative Assessment

Section 1:

Quiz 1.4-1, 1.4.2, 2.1-1, 2.1.2, 3.1, 3,2 and 3.3

Section 1:

Quiz 4, 5.1 and 6.1

Section 1:

Quiz 7.1 and 9.1

Summative Assessment

Section 1:

Project Unit 1-3_1 and Unit 1-3 Test

Section 2:

Project Unit 4-6_1 and Unit 4-6 Test

Section 3:

Project Unit 7-9_1 and Unit 7-9 Test

Internal Use Only

Pending Tasks

Fix the formatting

Break out into three separate optional lessons (each 5 sessions?)

Get Joe to create rubrics for mini-projects in each section

Future Work

Consider alternative assignments for hearing-impaired students. Music might be a very helpful tool for the majority of the hearing impaired. Perhaps of greater concern should be the visually impaired not just to this lesson but to all lessons in all courses.

(http://csmatters.org) 6 - 2

0b110 - 0b10

Optional

Data Visualization with Python and Bokeh

Unit 6. Data Visualization

Revision Date: Jul 18, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Summary

In this lesson, students will be formally introduced to data visualization using Bokeh: an interactive data visualization library in Python. They will go through a guided tour of how to use this library, and then will complete a fun activity that involves gathering data from their classmates for the purposes of visualization.

Outcomes

Students will learn why data visualization is important.

Students will be able to meaningfully visualize data they collect.

Overview

Session 1 (Line/Bar Graphs):

- Getting Started - Data acquisition (5 Minutes)
- Guided Tour - iPython notebook / Powerpoint - (15 minutes)
- Activity - Plotting data (30 minutes)
- Wrap Up - Exit slip (5 minutes)

Session 2 (High-level charts):

- Getting Started - Data acquisition (5 Minutes)
- Guided Tour - iPython notebook / Powerpoint (15 minutes)
- Activity - Plotting with high-level charts (30 minutes)
- Wrap Up - Exit slip (5 minutes)

Learning Objectives

CSP Objectives

- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
- 5.1.2 - Develop a correct program to solve problems.
- 5.3.1 - Use abstraction to manage complexity in programs.

Teacher Resources

Student computer usage for this lesson is: **required**

- Power Point presentations: "session1.pptx," "session2.pptx"
- iPython notebooks: "BokehTutorialLesson1_a.ipynb"

Lesson Plan

Session 1

Getting Started (5 minutes)

Come up with 4-5 questions that make sense when visualized with a line graph or bar graph, such as:

- When do you usually do your homework?
- What time do you wake up in the morning on a weekday?
- What time do you wake up in the morning on the weekend?
- etc.

Enlist the help of your students in coming up with the questions. Present all questions at the start of this lesson, and solicit answers to each question from each student. These answers will be used later in the lesson, so make sure to collect them in a table-like format such as ".csv"

It is reccommended that you create a Google Form to collect answers to these questions with. You could then ask your entire school to fill out the survey, and have even more data to visualize!

Guided Tour (15 Minutes)

Use the existing iPython Notebook to introduce Bokeh. Explain data visualization. Have the students follow along as you show them how Bokeh line plotting works, have them attempt the exercise at the end of the notebook.

Activity (30 Minutes)

Using the data that was collected at the beginning of class, determine the best way to plot the information collected at the beginning of class. Then, plot it! See if you can come up with cool trends by plotting data on the same chart, etc.

Exit Ticket (5 minutes)

Simple exit slip, ask the students what data visualization is, why it is important, how to construct a line graph in Bokeh, etc.

Session 2

Getting Started (5 minutes)

Gather data in a similar fashion as the last session. Ask questions of your class / school that make for interesting visualizations.

Guided Tour (15 Minutes)

Use the powerpoint provided to briefly discuss the high-level charts available in Bokeh.

Activity (30 Minutes)

Have the students use any of the high-level charts to plot the data that was collected in the warm-up.

Exit Ticket (5 minutes)

Ask a simple question about any of the plotting mechanisms covered.

Internal Use Only

Future Work

other session for working with larger data sets (csv files)

edit power points to be more extensive

(http://csmatters.org) 6 - 3

0b110 - 0b11

Optional

Dataquest

Unit 6. Data Visualization

Revision Date: Jul 18, 2016 (Version 2.0)
Duration: 15 50-minute sessions

Lesson Summary

Summary

Continuing the focus on data analysis from Unit Five, students will use the browser-based Dataquest learning environment (http://www.dataquest.io (http://www.dataquest.io)) and supplementary materials to explore more ways in which Python can be used to analyze data. For the first week, students will explore Dataquest through the browser-based "missions" on the website. Each lesson begins with a warm-up/journal entry, and students then spend the rest of the time working through the missions at their own pace. For the second part of the lesson, students will design and implement their own data analysis project in order to prepare them to complete a data-focused Create Performance task.

Outcomes

- Students will understand how to design a data analysis project
- Students will have the tools to analyze data in Python
- Students will have practice reading and understanding datasets

Overview

Week One: Learning Dataquest

1. Getting Started (5 - 10 min)
2. Independent Study (40 - 45 min)

Week Two: Data Analysis Project

1. Students Plan and Implement Data Analysis Program

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
  - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.

- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
  - 1.2.2B - A creative development process for creating computational artifacts can be used to solve problems when traditional or prescribed computing techniques are not effective.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4B - Effective collaborative teams consider the use of online collaborative tools.
  - 1.2.4C - Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation.
  - 1.2.4D - Effective collaboration strategies enhance performance.
  - 1.2.4E - Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.
- 2.2.1 - Develop an abstraction when writing a program or creating other computational artifacts.
- 3.1.1 - Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.
- 3.1.2 - Collaborate when processing information to gain insight and knowledge.
- 3.1.3 - Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.
- 3.2.1 - Extract information from data to discover and explain connections or trends
- 3.2.2 - Determine how large data sets impact the use of computational processes to discover information and knowledge.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
  - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
  - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
  - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
  - 4.1.1H - Different algorithms can be developed to solve the same problem.
  - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.
  - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
  - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
  - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
  - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
  - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
  - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
  - 5.1.1C - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution.
  - 5.1.1D - Additional desired outcomes may be realized independently of the original purpose of the program.
- 5.1.2 - Develop a correct program to solve problems.
  - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
  - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
  - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
  - 5.1.2D - Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.
  - 5.1.2E - Documentation about program components, such as code segments and procedures, helps in developing and maintaining programs.
  - 5.1.2F - Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments.
  - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.
  - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.1.3 - Collaborate to develop a program.
  - 5.1.3A - Collaboration can decrease the size and complexity of tasks required of individual programmers.
  - 5.1.3B - Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.
  - 5.1.3C - Collaboration in the iterative development of a program requires different skills than developing a program alone.
  - 5.1.3D - Collaboration can make it easier to find and correct errors when developing programs.
  - 5.1.3E - Collaboration facilitates developing program components independently.
  - 5.1.3F - Effective communication between participants is required for successful collaboration when developing programs.
- 5.2.1 - Explain how programs implement algorithms.
  - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
  - 5.2.1B - Program instructions are executed sequentially.
  - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
  - 5.2.1E - Program execution automates processes.
  - 5.2.1I - Executable programs increase the scale of problems that can be addressed.
- 5.4.1 - Evaluate the correctness of a program.
  - 5.4.1A - Program style can affect the determination of program correctness.
  - 5.4.1B - Duplicated code can make it harder to reason about a program.
  - 5.4.1C - Meaningful names for variables and procedures help people better understand programs.

- 5.4.1D - Longer code segments are harder to reason about than shorter code segments in a program.
- 5.4.1E - Locating and correcting errors in a program is called debugging the program.
- 5.4.1G - Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
- 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
- 5.4.1I - Programmers justify and explain a program's correctness.
- 5.4.1J - Justification can include a written explanation about how a program meets its specifications.
- 5.4.1K - Correctness of a program depends on correctness of program components, including code segments and procedures.
- 5.4.1L - An explanation of a program helps people understand the functionality and purpose of it.
- 5.4.1M - The functionality of a program is often described by how a user interacts with it.
- 5.4.1N - The functionality of a program is best described at a high level by what the program does, not at the lower level of how the program statements work to accomplish this.

- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
  - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
  - 5.5.1B - Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.
  - 5.5.1C - Real numbers are approximated by floating-point representations that do not necessarily have infinite precision.
  - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
  - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
  - 5.5.1F - Compound expressions using and, or, and not are part of most programming languages.
  - 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.

Teacher Resources

Student computer usage for this lesson is: **required**

DataQuest.io website: https://www.dataquest.io/learn (https://www.dataquest.io/learn)

Week One Materials: **Unit 6 Resources -> DataQuest.io -> Week One Lesson Materials -> Mission #**

Week Two Materials:

Datasets: **Unit 6 Resources -> Dataquest.io - > Week Two Project Datasets and Materials -> Datasets**

Sample Project: **Unit 6 Resources -> Dataquest.io - > Week Two Project Datasets and Materials -> Sample Project**

Project Rubric: **Unit 6 Resources -> Dataquest.io - > Week Two Project Datasets and Materials -> "Data Analysis Project Rubric"**

(Quizzes for Week One and Week Two are in the corresponding teacher-only resource folders)

Lesson Plan

Week One: Learning Dataquest

Note: all worksheets and quizzes can be found in the teacher-only resource folder, Unit Six -> DataQuest.io -> Week One Lesson Materials -> Mission #

Directions for working in Dataquest.io

1. Each student will first need to create an account on Dataquest.io. This is free, and will help them to keep track of their progress.
2. Each mission comes with a worksheet with required sections to complete. Students are encouraged to fill out as much as possible. The non-required sections are introductions to basic coding tools. Some students may want to do these if they need a refresher on the concepts.
   - **Note:** As of now, sections cannot be skipped on the website. This limitation may change in the future.
3. Once they have completed the worksheet for the mission, students will use the notes on their worksheets to complete:
   a. A **concept quiz** to test their understanding of the data science concepts.
   b. A **coding quiz** to test their understanding of the Python concepts.

Quizzes should be done in class, and should take a minimum of 10-20 minutes to complete. It is advisable to not give a quiz out in the last ten minutes of class. If there are only a few minutes left, the student can use the time to add to their notes.

If a student fails one of the quizzes, they may be given the chance to go back and add to their worksheet before attempting the quiz again. (Multiple versions of all coding quizzes are available.)

There are a total of four Missions for the Introduction to Python track. Students are only required to do the first three. The fourth Mission has worksheets/quizzes for those students who get to it, and can be counted as extra credit/normal grade at the teacher's discretion. Two additional optional missions are available: one on data visualization, and one on working with statistics.

Getting Srarted (5 - 10 min)

Day 1

Show the first two minutes of the introductory video in Mission 1 on the Dataquest.io website. Students will discuss their reactions and thoughts about Data Science.

Day 2

Pull up d3js.org (http://d3js.org) on the projector. This is the webpage for a data visualization library in Javascript that has many great examples of ways to make connections from data. You can explore by clicking on one of the tiles on the front page. Explore the D3 front page with the class and discuss reactions.

Day 3

This warm-up time is used for class discussion on progress through the missions. You can use this time to gauge the students' comfort with Python concepts by having students vote with their heads down. If enough students are having trouble, you may want to have a separate review session during the class.

Day 4

This warm-up time can be used for reviewing a Python concept (such as Dictionaries) or looking at a current news article involving data analysis (any article about a topic of interest to the students that uses statistics would be appropriate). Students should think-pair-share on additional ways in which data could be used.

Day 5

Students should do a show of hands to see where everyone is in the missions. The class should have a general discussion about progress.

Week Two: Data Analysis Project

Note: All materials for this section can be found in Unit 6 Resources -> Dataquest.io - > Week Two Project Datasets and Materials

Directions:

For this week, students will be pairing up to create and implement a data analysis program of their own design.

- Teachers start out the first day by presenting the PowerPoint "Project Introduction," which goes through the steps to creating a data analysis project. Teachers then review the "Data Analysis Project Directions" document.
- The class splits up into groups of two (with up to one group of three) and each group chooses a dataset to work with. It is preferable if each group chooses a different dataset.

For the rest of the week, students should work on their projects in their groups. At the end, teachers can optionally have them present their PowerPoints to the class, exchange presentations in pairs, or merely turn everything in.

Evidence of Learning

Formative Assessment

- Week One quizzes
- Check for understanding at the beginning of each day of week one.

Summative Assessment

Week two project.

(http://csmatters.org) 6 - 4

0b110 - 0b100

Optional

Diversity Makes For Better Solutions

Unit 6. Data Visualization

Revision Date: Sep 07, 2016 (Version 2.0)
Duration: 2 50-minute sessions

Lesson Summary

Pre-lesson preparation

The second session is a programming activity and requires using NetLogo (a tutorial is in the Unit 4 Lesson on Hypothesis Testing with NetLogo) or using Python with Bokeh (for more advanced students).

Summary

Students will be introduced to the topic of diversity and computational problems, learn about and discuss implications of the proliferation of algorithms in society, and reflect on how the quality of input data and small, sometimes unintended biases can lead to low quality solutions.

Outcomes

- Students will consider how diversity leads to better solutions.
- Students will discuss the proliferation of algorithms in society.
- Students will learn about some applications of artificial intelligence and machine learning algorithms.
- Students will explore a computational simulation of bias.
- Students will understand how small changes in input parameters can affect output quality and variety.
- Students will program an agent-based model from social science theory.

Overview

Session 1

1. Getting Started (5 min)
    - Artificial intelligence algorithms in daily life?
2. Discussion (10 min)
    - Algorithms, Input Data, and Diversity
3. Guided Activty (25 min)
    - Parable of the Polygons
4. Wrap-up (10 min)

Session 2

1. Getting Started (5 min)
    - Algorithms, Diversity, and Solutions
2. Programming Activity (40 min)
    - Make your own Parable of the Polygons
3. Wrap-up (5 min)

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
    - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
    - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
    - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
    - 1.2.2B - A creative development process for creating computational artifacts can be used to solve problems when traditional or prescribed computing techniques are not effective.
- 1.2.3 - Create a new computational artifact by combining or modifying existing artifacts.
    - 1.2.3A - Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
- 1.2.5 - Analyze the correctness, usability, functionality, and suitability of computational artifacts.
    - 1.2.5A - The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact.
    - 1.2.5D - The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.
- 2.3.1 - Use models and simulations to represent phenomena.

    - 2.3.1A - Models and simulations are simplified representations of more complex objects or phenomena.
    - 2.3.1B - Models may use different abstractions or levels of abstraction depending on the objects or phenomena being posed.
    - 2.3.1C - Models often omit unnecessary features of the objects or phenomena that are being modeled.
    - 2.3.1D - Simulations mimic real-world events without the cost or danger of building and testing the phenomena in the real world.
- 2.3.2 - Use models and simulations to formulate, refine and test hypotheses.
    - 2.3.2A - Models and simulations facilitate the formulation and refinement of hypotheses related to the objects or phenomena under consideration.
    - 2.3.2D - The results of simulations may generate new knowledge and new hypotheses related to the phenomena being modeled.
    - 2.3.2F - Simulations can facilitate extensive and rapid testing of models.
- 3.1.1 - Find patterns and test hypotheses about digitally processed information to gain insight and knowledge.
    - 3.1.1A - Computers are used in an iterative and interactive way when processing digital information to gain insight and knowledge.
    - 3.1.1E - Patterns can emerge when data is transformed using computational tools.
- 3.1.3 - Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notation and precise language.
    - 3.1.3A - Visualization tools and software can communicate information about data.
    - 3.1.3B - Tables, diagrams, and textual displays can be used in communicating insight and knowledge gained from data.
    - 3.1.3C - Summaries of data analyzed computationally can be effective in communicating insight and knowledge gained from digitally represented information.
    - 3.1.3E - Interactivity with data is an aspect of communicating.
- 3.2.1 - Extract information from data to discover and explain connections or trends
    - 3.2.1G - Metadata is data about data.
- 7.1.1 - Explain how computing innovations affect communication, interaction and cognition.
    - 7.1.1E - Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results.
    - 7.1.1G - Search trends are predictors.
    - 7.1.1O - The Internet and the Web have impacted productivity, positively and negatively, in many areas.
- 7.2.1 - Explain how computing has impacted innovations in other fields.
    - 7.2.1A - Machine learning and data mining have enabled innovation in medicine, business, and science.
- 7.3.1 - Analyze the beneficial and harmful effects of computing.
    - 7.3.1A - Innovations enabled by computing raise legal and ethical concerns.
    - 7.3.1D - Both authenticated and anonymous access to digital information raise legal and ethical concerns.
    - 7.3.1E - Commercial and governmental censorship of digital information raise legal and ethical concerns.
    - 7.3.1G - Privacy and security concerns arise in the development and use of computational systems and artifacts.
    - 7.3.1H - Aggregation of information, such as geolocation, cookies, and browsing history, raises privacy and security concerns.
    - 7.3.1J - Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.
    - 7.3.1L - Commercial and governmental curation of information may be exploited if privacy and other protections are ignored.
    - 7.3.1M - Targeted advertising is used to help individuals, but it can be misused at both individual and aggregate levels.
- 7.4.1 - Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts.
    - 7.4.1C - The global distribution of computing resources raises issues of equity, access, and power.
    - 7.4.1D - Groups and individuals are affected by the "digital divide" - differing access to computing and the Internet based on socioeconomic or geographic characteristics.

Math Common Core Practice:

- MP3: Construct viable arguments and critique the reasoning of others.

Common Core ELA:

- RST 12.2 - Determine central ideas and conclusions in the text
- RST 12.6 - Analyze the author's purpose in providing an explanation, describing a procedure
- RST 12.9 - Synthesize information from a range of sources

NGSS Practices:

- 2. Developing and using models
- 7. Engaging in argument from evidence

NGSS Content:

- HS-ETS1-1. Analyze a major global challenge to specify qualitative and quantitative criteria and constraints for solutions that account for societal needs and wants.
- HS-ETS1-4. Use a computer simulation to model the impact of proposed solutions to a complex real-world problem with numerous criteria and constraints on interactions within and between systems relevant to the problem.

Teacher Resources

Student computer usage for this lesson is: **required**

Supplementary articles:

- "How Diversity Makes Us Smarter"
  - http://www.scientificamerican.com/article/how-diversity-makes-us-smarter/ (http://www.scientificamerican.com/article/how-diversity-makes-us-smarter/)

Lesson Plan

Session 1

Getting Started (5 min)

Think-Pair-Share: Artificial intelligence algorithms in daily life?

Ask your students to list discuss what artificial intelligence algorithms they think affect them in daily life and how they are affected. How would it be different if those algorithms did not recognize them or treated them inappropriately? If an algorithm makes the wrong judgement about them, what would they do, who could they go to for help? It is OK if students are unsure about the answers as it leads into the following discussion.

Discussion: Algorithms, Input Data, and Diversity (10 min)

Algorithms have become prominent and common in daily life. In particular, algorithms that use **artificial intelligence (AI)** or a technique called **machine learning** are employed to make decisions that affect us, sometimes in ways we aren't aware of and cannot control. For example, machine learning is used by Google, Facebook, Flickr, among other companies to detect faces in photographs, apply labels, or automatically tag images with people they recognize. In general, these algorithms are *trained* on input data (such as images of people), and are improved iteratively using statistical methods until they learn how to categorize, differentiate, and reach a correct answer. Another example of companies using AI is in personalized marketing or recommendation systems. For instance, Amazon tailors the products they suggest based on an algorithm trained on similar customers' purchase history. Or a company like Netflix will recommend movies to users who rate similar movies similarly.

At the heart of these algorithms is the fact that their solutions all depend on the initial parameters and input data used to train them. What happens when those parameters and data are not sufficiently diverse? Small changes in those parameters, oversights in the training data, implementer bias (both intended and not), and missing distributions or categories can all negatively prejudice algorithms. All this is important to keep in mind as algorithms start determining aspects of our lives and there is less or no control over the input and decisions they make about and for us.

A 2016 article in the New York Times (http://www.nytimes.com/2016/06/26/opinion/sunday/artificial-intelligences-white-guy-problem.html?_r=0 (http://www.nytimes.com/2016/06/26/opinion/sunday/artificial-intelligences-white-guy-problem.html?_r=0)) highlights an example problem where lack of diversity leads to lower quality solutions. In particular, various AI and machine learning algorithms were developed by and calibrated on too limited a population, leading to insulting and significantly bad results, such as not recognizing people with dark skin tones or tagging them inappropriately. Examples where diversity may greatly influence the quality and real-world implications of algorithms include:

- Software used by police for assessing risk has an unwarranted racial bias (https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing (https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing))
- Targeting and financial exploitation of more vulnerable communities (https://bigdata.fairness.io/data-brokers/ (https://bigdata.fairness.io/data-brokers/))
- Hiring algorithms and software like E-verify, a program to determine legal eligibility for the workforce (https://bigdata.fairness.io/e-verify/ (https://bigdata.fairness.io/e-verify/))
- Online searching and results that are returned (https://bigdata.fairness.io/search-bias/ (https://bigdata.fairness.io/search-bias/))
- Credit scoring algorithms (https://www.thenation.com/article/how-companies-turn-your-facebook-activity-credit-score/ (https://www.thenation.com/article/how-companies-turn-your-facebook-activity-credit-score/))
- Predictive policing and surveillance, such as the NSA's alleged leaked program XKEYSCORE, tracks metrics and metadata about internet users, potentially then fed into various other surveillance apparatuses (https://theintercept.com/2015/07/01/nsas-google-worlds-private-communications/ (https://theintercept.com/2015/07/01/nsas-google-worlds-private-communications/))
- In 2014, the White House publicly recognized that "Big Data" algorithms may foster discrimination and that the federal government would need to begin being more watchful of such practices (https://www.whitehouse.gov/sites/default/files/docs/big_data_privacy_report_may_1_2014.pdf (https://www.whitehouse.gov/sites/default/files/docs/big_data_privacy_report_may_1_2014.pdf))
- In 2016, the FTC published a report specifically on the potential of such algorithms to foster both inclusivity and exclusivity (https://www.ftc.gov/system/files/documents/reports/big-data-tool-inclusion-or-exclusion-understanding-issues/160106big-data-rpt.pdf (https://www.ftc.gov/system/files/documents/reports/big-data-tool-inclusion-or-exclusion-understanding-issues/160106big-data-rpt.pdf))

Guided Activity (25 min)

Parable of the Polygons

A hands-on example of how initial parameters, subtle biases, and lack of diversity can greatly affect an algorithm's execution is in the "Parable of the Polygons" (http://ncase.me/polygons/ (http://ncase.me/polygons/)). The Parable of the Polygon is a simulation based on the mathematical model of how segregated communities may arise from small differences preferences, based on the theory of Thomas Schelling, an economist and

game theorist who received the Nobel Prize for Economics in 2005.

Guide students through the website and let them explore dragging polygons, changing settings, and see how they can influence (it is best for them to follow along individually or in groups on their own computer).

An alternative version of the final sandbox with three polygon shapes is also available here: http://ncase.me/polygons-pentagons/play/automatic/automatic_sandbox_frame.html (http://ncase.me/polygons-pentagons/play/automatic/automatic_sandbox_frame.html)

Wrap-up (10 min)

Journal or Homework

Have your students continue to explore the Parable of the Polygons website and answer the following questions using the sandbox at the bottom of the website:

- What settings where the ratio of polygon populations are the same lead to the greatest segregation? Why do you think that is?
- What settings where the ratio of polygon populations are the same lead to the least segregation? Why is that?
- What settings where polygons still have room to move lead to the algorithm being unable to find a solution (it does not terminate)? Why is that?
- Answer the same three questions above using the three-polygon version here: http://ncase.me/polygons-pentagons/play/automatic/automatic_sandbox_frame.html (http://ncase.me/polygons-pentagons/play/automatic/automatic_sandbox_frame.html)

Session 2

Getting Started (5 Min)

Think-Pair-Share: Algorithms, Diversity, and Solutions [5 min]

- Have students consider one or more of the following questions:
  - How can an algorithm be affected by diversity (such as in its input data)?
  - What does it mean for input or training data to be diverse?
  - Can you think of examples where diversity, or the lack of diversity, affects the quality of an algorithm's solution?
  - Can you think of algorithms that benefit from diverse data?

Programming Activity (40 Min)

Two options: guided programming exploration with NetLogo, or more involved Python programming project for advanced students

Option 1: Using NetLogo

- If you have not already used NetLogo in the classroom, you can follow the instructions for downloading and setting up NetLogo in the Unit 4 Lesson "Hypothesis Testing with Simulations in NetLogo"
- Open up NetLogo and go to File->Models Library, then under the Sample Models folder, open the Social Science folder and load the Segregation model
- For information about the model, read the sections in the Info tab
- Have the students experiment with the settings and see what differences and similarities it has to the Parable of the Polygons website
- For the remainder of the activity, have the students make some meaningful changes to the code of the simulation (in the Code tab) and write a short response stating what they did and how their parameters
- Some suggestions include:
  - Add more types of agents to the model (hint, add another color to the line "set color one-of [red green]")
  - Change the sliders to match the Parable of the Polygons slider (agent count based on ratios, ability to have <50% agents)
- Finally, a more developed model is included in the NetLogo models library

Option 2: Using Python with Bokeh **(for advanced students)**

Have students re-create the Parable of the Polygons themselves in Python, time with at least four types of polygon shape/colors. The source code for the Parable website is hosted on GitHub here: https://github.com/ncase/polygons/blob/gh-pages/play/automatic/automatic.js (https://github.com/ncase/polygons/blob/gh-pages/play/automatic/automatic.js) It is in JavaScript, but it serves as a good pseudocode basis. Alternatively, students could use the NetLogo code from Option 1 as another reference in creating their simulation.

Wrap-up (5 Min)

Journal: Have your students reflect and list 5 examples or ways in which diverse input makes for better solutions.

(http://csmatters.org) C - 1

0bC - 0b1

Create - Application from Ideas

Unit Create Performance Task

Revision Date: Jul 18, 2016 (Version 2.0)
Duration: 15 50-minute sessions

Lesson Summary

Pre-lesson Preparation

Students should have practice in the following areas before starting on the Create Performance Task:

- Designing a program using multiple-level algorithms (at different levels of abstraction) and data abstractions, and being able to identify how they are used.
- Creating a program with the following features:
  - Mathematical expressions.
  - Logical expressions.
  - Sequence, selection, and iteration.
  - Functions.
- Creating required submission documents, including:
  - Creating a video (narration optional) that illustrates the functionality of a program.
  - Saving program code as a PDF.
  - Marking segments of the code using ovals and squares to signify algorithms and the use of abstraction.

Summary

In the Create Performance Task (CPT), students bring ideas to life through software development. The CPT requires students to design and iteratively develop a program. The College Board encourages but does not require students to collaborate with a partner, and once students begin the development of the CPT, they may receive help only from the collaborative partner. While not requiring it, this lesson supports a collaborative effort. Even if collaboration is used, each student must independently complete a significant level of planning, designing, and developing the program. Students will have 12 hours of class time to complete, and submit:

- A video of the program running.
- Written responses about the program and the development process.
- All of their program code.

The program must:

1. Produce a result that cannot easily be accomplished without computing tools.
2. Demonstrate a variety of capabilities.
3. Include an algorithm that integrates other algorithms and mathematical or logical concepts.
4. Use abstraction (functions or data structures) to manage the complexity of the program.
5. Implement several language features and use the following concepts:
   - Mathematical expressions such as numbers, variables, and arithmetic operators.
   - Logical expressions using Boolean operators.
   - Decision or selection branches.
   - Iteration or looping.
   - Collections such as strings or lists.

This lesson plan provides a schedule for 15 50-minute sessions to meet the 12-hour in-class minimum required by the College Board.

Outcomes

- Students will design and implement a program to pursue a topic of personal interest or to solve a problem.
- Students will use functional or data abstraction to manage program complexity.
- Students will be able to implement a multiple-level algorithm.
- Students will be able to implement sequence, selection, iteration, and functions using arithmetic and logical expressions.

Overview

Session 1: Identify performance task requirements and choose a topic to develop.

Session 2: Design the program, identifying both multiple -evel algorithms and multiple abstractions used in their design. Students work with collaborative partners to verify the adequacy of their designs to meet the performance task requirements.

Session 3: Program Development: Code top-level functions.

Session 4: Program Development: Code second-level functions.

Session 5: Program Development Checkpoint: Assess progress and revise design and development strategies.

Session 6: Program Development: Continue revised function development.

Session 7: Program Development: Complete function development.

Session 8: Program Development Checkpoint: Assess progress and make final revision to development strategies.

Session 9: Program Development: Make final revisions to program code.

Session 10: Program Development: Complete program and plan video.

Session 11: Video Development: Record and optionally narrate video.

Session 12: Video Development: Assess video and make final revisions.

Session 13: Written Responses: Respond to prompts 2a, 2b, and 2c.

Session 14: Written Responses: Respond to prompts 2d and 2e.

Session 15: CPT Submission: Submit and share projects.

Learning Objectives

CSP Objectives

- 1.1.1 - Apply a creative development process when creating computational artifacts.
  - 1.1.1A - A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.
  - 1.1.1B - Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
- 1.2.1 - Create a computational artifact for creative expression.
  - 1.2.1A - A computational artifact is something created by a human using a computer and can be, but is not limited to, a program, an image, an audio, a video, a presentation, or a Web page file.
  - 1.2.1B - Creating computational artifacts requires understanding of and use of software tools and services.
  - 1.2.1E - Creative expressions in a computational artifact can reflect personal expressions of ideas or interests.
- 1.2.2 - Create a computational artifact using computing tools and techniques to solve a problem.
  - 1.2.2A - Computing tools and techniques can enhance the process of finding a solution to a problem.
  - 1.2.2B - A creative development process for creating computational artifacts can be used to solve problems when traditional or prescribed computing techniques are not effective.
- 1.2.4 - Collaborate in the creation of computational artifacts.
  - 1.2.4A - A collaboratively created computational artifact reflects effort by more than one person.
  - 1.2.4C - Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation.
  - 1.2.4D - Effective collaboration strategies enhance performance.
  - 1.2.4E - Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.
  - 1.2.4F - A collaboratively created computational artifact can reflect personal expressions of ideas.
- 1.3.1 - Use computing tools and techniques for creative expression.
  - 1.3.1B - Digital audio and music can be created by synthesizing sounds, sampling existing audio and music, and recording and manipulating sounds, including layering and looping.
  - 1.3.1C - Digital images can be created by generating pixel patterns, manipulating existing digital images, or combining images.
  - 1.3.1D - Digital effects and animations can be created by using existing software or modified software that includes functionality to implement the effects and animations.
  - 1.3.1E - Computing enables creative exploration of both real and virtual phenomena.
- 2.1.1 - Describe the variety of abstractions used to represent data.
- 2.2.1 - Develop an abstraction when writing a program or creating other computational artifacts.
- 2.2.2 - Use multiple levels of abstraction to write programs.
- 2.2.3 - Identify multiple levels of abstractions used when writing programs.
- 4.1.1 - Develop an algorithm for implementation in a program.
  - 4.1.1A - Sequencing, selection, and iteration are building blocks of algorithms.
  - 4.1.1B - Sequencing is the application of each step of an algorithm in the order in which the statements are given.
  - 4.1.1C - Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
  - 4.1.1D - Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
  - 4.1.1E - Algorithms can be combined to make new algorithms.
  - 4.1.1F - Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.
  - 4.1.1G - Knowledge of standard algorithms can help in constructing new algorithms.
  - 4.1.1H - Different algorithms can be developed to solve the same problem.
  - 4.1.1I - Developing a new algorithm to solve a problem can yield insight into the problem.
- 4.1.2 - Express an algorithm in a language.

    - 4.1.2A - Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
    - 4.1.2B - Natural language and pseudocode describe algorithms so that humans can understand them.
    - 4.1.2C - Algorithms described in programming languages can be executed on a computer.
    - 4.1.2D - Different languages are better suited for expressing different algorithms.
    - 4.1.2G - Every algorithm can be constructed using only sequencing, selection, and iteration.
    - 4.1.2I - Clarity and readability are important considerations when expressing an algorithm in a language.
- 5.1.1 - Develop a program for creative expression to satisfy personal curiosity or to create new knowledge.
    - 5.1.1A - Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.
    - 5.1.1B - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.
    - 5.1.1C - Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution.
    - 5.1.1D - Additional desired outcomes may be realized independently of the original purpose of the program.
    - 5.1.1E - A computer program or the results of running a program may be rapidly shared with a large number of users and can have widespread impact on individuals, organizations, and society.
- 5.1.2 - Develop a correct program to solve problems.
    - 5.1.2A - An iterative process of program development helps in developing a correct program to solve problems.
    - 5.1.2B - Developing correct program components and then combining them helps in creating correct programs.
    - 5.1.2C - Incrementally adding tested program segments to correct working programs helps create large correct programs.
    - 5.1.2D - Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.
    - 5.1.2E - Documentation about program components, such as code segments and procedures, helps in developing and maintaining programs.
    - 5.1.2F - Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments.
    - 5.1.2G - Program development includes identifying programmer and user concerns that affect the solution to problems.
    - 5.1.2I - A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.
    - 5.1.2J - A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
- 5.1.3 - Collaborate to develop a program.
    - 5.1.3A - Collaboration can decrease the size and complexity of tasks required of individual programmers.
    - 5.1.3B - Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.
    - 5.1.3C - Collaboration in the iterative development of a program requires different skills than developing a program alone.
    - 5.1.3D - Collaboration can make it easier to find and correct errors when developing programs.
    - 5.1.3E - Collaboration facilitates developing program components independently.
    - 5.1.3F - Effective communication between participants is required for successful collaboration when developing programs.
- 5.2.1 - Explain how programs implement algorithms.
    - 5.2.1A - Algorithms are implemented using program instructions that are processed during program execution.
    - 5.2.1B - Program instructions are executed sequentially.
    - 5.2.1C - Program instructions may involve variables that are initialized and updated, read, and written.
    - 5.2.1E - Program execution automates processes.
    - 5.2.1F - Processes use memory, a central processing unit (CPU), and input and output.
    - 5.2.1I - Executable programs increase the scale of problems that can be addressed.
    - 5.2.1J - Simple algorithms can solve a large set of problems when automated.
    - 5.2.1K - Improvements in algorithms, hardware, and software increase the kinds of problems and the size of problems solvable by programming.
- 5.3.1 - Use abstraction to manage complexity in programs.
    - 5.3.1A - Procedures are reusable programming abstractions.
    - 5.3.1B - A procedure is a named grouping of programming instructions.
    - 5.3.1C - Procedures reduce the complexity of writing and maintaining programs.
    - 5.3.1D - Procedures have names and may have parameters and return values.
    - 5.3.1E - Parameterization can generalize a specific solution.
    - 5.3.1F - Parameters generalize a solution by allowing a procedure to be used instead of duplicated code.
    - 5.3.1G - Parameters provide different values as input to procedures when they are called in a program.
    - 5.3.1H - Data abstraction provides a means of separating behavior from implementation.
    - 5.3.1I - Strings and string operations, including concatenation and some form of substring, are common in many programs.
    - 5.3.1J - Integers and floating-point numbers are used in programs without requiring understanding of how they are implemented.
    - 5.3.1K - Lists and list operations, such as add, remove, and search, are common in many programs.
    - 5.3.1L - Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.
    - 5.3.1M - Application program interfaces (APIs) and libraries simplify complex programming tasks.
    - 5.3.1N - Documentation for an API/library is an important aspect of programming.
    - 5.3.1O - APIs connect software components, allowing them to communicate.
- 5.4.1 - Evaluate the correctness of a program.
    - 5.4.1A - Program style can affect the determination of program correctness.
    - 5.4.1B - Duplicated code can make it harder to reason about a program.
    - 5.4.1C - Meaningful names for variables and procedures help people better understand programs.
    - 5.4.1D - Longer code segments are harder to reason about than shorter code segments in a program.
    - 5.4.1E - Locating and correcting errors in a program is called debugging the program.

- 5.4.1H - Visual displays (or different modalities) of program state can help in finding errors.
- 5.4.1I - Programmers justify and explain a program's correctness.
- 5.4.1J - Justification can include a written explanation about how a program meets its specifications.
- 5.4.1K - Correctness of a program depends on correctness of program components, including code segments and procedures.
- 5.4.1L - An explanation of a program helps people understand the functionality and purpose of it.
- 5.4.1M - The functionality of a program is often described by how a user interacts with it.
- 5.4.1N - The functionality of a program is best described at a high level by what the program does, not at the lower level of how the program statements work to accomplish this.

- 5.5.1 - Employ appropriate mathematical and logical concepts in programming.
  - 5.5.1A - Numbers and numerical concepts are fundamental to programming.
  - 5.5.1D - Mathematical expressions using arithmetic operators are part of most programming languages.
  - 5.5.1E - Logical concepts and Boolean algebra are fundamental to programming.
  - 5.5.1F - Compound expressions using and, or, and not are part of most programming languages.
  - 5.5.1G - Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.
  - 5.5.1H - Computational methods may use lists and collections to solve problems.
  - 5.5.1I - Lists and other collections can be treated as abstract data types (ADTs) in developing programs.
  - 5.5.1J - Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection.

Essential Questions

- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How are algorithms implemented and executed on computers and computational devices?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

In the Lesson Resources Folder (Teacher Only)

- Create Performance Task Rubric as of November 2015

In the Lesson Resources Folder (Public)

- Preliminary Topic Selection Guide
- Create Performance Task Rubric Assignment
- Create Performance Task Directions as of November 2015
- Daily Progress Report
- Collaboration Agreement
- Program Development Plan

Possible Screen Capture/Recording Tool

VLC Media Player

Dowload: http://www.videolan.org/vlc/index.html (http://www.videolan.org/vlc/index.html)

Instructions for screen recording: http://www.wikihow.com/Screen-Capture-to-File-Using-VLC (http://www.wikihow.com/Screen-Capture-to-File-Using-VLC)

Lesson Plan

Teacher Guidelines for Involvement

Prior to administration of the Performance Task and during the planning stages:

Teachers MAY:

- Influence student selection of topics.
- Clarify directions for completion of the performance tasks.
- Remind students of submission requirements.
- Guide student understanding of the CPT.
- Designate collaborative partners

- **Influence** student selection of program topics (however, teachers may not require selection of particular topics for students).
- Assist students in creating a program development schedule.

Teachers MAY NOT:

- Choose topics for students.
- Develop, write, revise, or amend student work.
- Provide boilerplates.
- Allow students to submit practice tasks as actual performance tasks.
- Suggest or evaluate student responses to prompts.

During the Performance Task

Teachers MAY:

- Resolve hardware/technical problems.
- Manage student progress throughout the task.
- Assist students in managing their progress, including making assessments of student efforts and fulfillment of the development schedule
- Clarify directions for the performance tasks.
- Remind students of submission requirements.

Teachers MAY NOT:

- Write, revise, or amend student work.
- Provide boilerplates.
- Allow students to submit practice tasks as actual performance tasks.
- Provide programming support while students are developing the performance task. **Only code written by the student will be scored.**
- Provide feedback to students regarding the quality of the work until after the task has been finally submitted to the College Board.

Session 1

Objectives:

- Introduce the Create Performance Task (CPT).
- Students develop understanding of the CPT requirements.
- Students assess both the performance task requirements and its evaluation rubric.
- Students are designated partners and do preliminary topic selection.

Tell students: Today you will begin the Create Performance Task. In the CPT, you will be developing a program of your choice. The goal of this task is for you to develop a program to solve a problem, create an innovation, or express a personal interest.

Exercise 1: CPT Overview

Provide students with a copy of the Create Performance Task (AP Computer Science Principles Performance Task Create–Applications from Ideas) description from the College Board.

1. Students read the Overview section on page 1 and answer the three questions below. (2 min)
    a. How is the program topic determined?
    b. What three products are you to produce?
    c. How much time will you have to complete and submit the required products?
2. After students share with elbow partners, address any questions students have to this point. (3 min)
3. Students read the General Requirements section beginning on page 1 and answer the questions below. (5 min)
    a. What recommendation regarding collaboration is made?
    b. What are the five general requirements for this performance task?
    c. What does the video need to display?
    d. What are the written responses about?

Exercise 2: Program Requirements

1. Students read Program Requirements section. (1 min)
2. They then answer the questions below: (5 min)
    a. What must the program demonstrate?
    b. What must the program implement?
    c. What must the program produce?
    d. What five elements from the second sentence should the program contain?
    e. What must the program demonstrate about each of the following?
        - Mathematical and logical concepts
        - Implementation of an algorithm
        - Use of abstractions
3. After students share with elbow partners, address any questions that students have to this point. (3 min)

Exercise 3: Submission Requirements

1. Read part 1 of the Submission Requirements about the video and answer the following. (2 min)
    a. What three requirements does the video have to meet?
2. Read the first paragraph of part 2 about the Written Responses and answer the following. (2 min)
    a. What format is to be used for the written responses?
    b. How are the responses to be labeled?
    c. What is the maximum length of all written response combined?

Exercise 4: Written Response

Jigsaw the Program Purpose and Development section. Assign each of the written responses a-e to one of five groups. Each group answers the questions below. (2 min)

Written Response a

1. What must the response identify?
2. What must the response explain?
3. What is the approximate word limit?

Written Response b

1. What is focused on in this description?
2. In addition to describing the difficulties, what must be described about them?
3. At least one of the difficulties must have been addressed in what manner?
4. What is the approximate word limit?

Written Response c

1. What portion of the program code is to be copied in this section?
2. What is to be described?
3. What is the approximate word limit?

Written Response d

1. What portion of the program code is to be copied in this section?
2. What is to be explained?
3. What is the approximate word limit?

Written Response e

1. What is contained in this response?
2. What is to be marked with an oval?
3. What is to be marked with a rectangle?
4. When are comments are required?

Each group shares its results. (5 min)

Exercise 5:

Assign students collaborative partners. Distribute *Preliminary Topic Selection Guide*. Students decide on the topic they will address and the form of collaboration they will use. (10 min)

Homework:

Distribute the CPT rubric and CPT Rubric Assignment.

Session 2

Objectives:

- Students commit to a program topic.
- Students plan collaboration strategy.
- Students design the program with multiple-level algorithms and multiple abstractions.
- Students develop a formal schedule for program implementation.

Tell students:

Today we have four goals. We will commit to a program topic and develop a collaboration strategy, a program design, and an implementation schedule.

NOTE: Students may only receive help from the collaborative partner from this point on until the CPT documents are ready to be submitted. Teachers collect the topics, strategies, and designs in order to help manage student progress, but may not provide any feedback to students on content.

Exercise 1: Program Topic (5 min)

Meet with collaborative partners and discuss the CPT rubric assignment. Each individual submits programming topic and purpose.

Exercise 2: Collaboration Strategy (10 min)

Each partner group submits a plan for collaboration.

Exercise 3: Program Design (20 min)

Each individual submits a program design.

Exercise 4: Program Development Schedule (10 min)

Each individual submits a daily program development schedule.

Session 3: Program Development Day 1

Tell students: Today will be our first program design day.

Exercise: Students implement their program (48 min)

At the end of class:

- Students complete a daily progress report.
- Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content. (2 min)

Session 4: Program Development Day 2

Tell students: Today will be our second program design day.

Exercise: Students implement their program (48 min)

At the end of class:

- Students complete a daily progress report.
- Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content. (2 min)

Session 5: First Checkpoint

Tell students: Today we will evaluate program development progress and revise our program development schedules as needed. We will also discuss with partners the difficulties we encountered and the strategies we used to address them.

Note: Partners collaborating in their program development in the first session stages will switch to individual development at this point.

Exercise 1: Review Program Development Progress (5 min)

Each individual submits a revised schedule of program development,

Exercise 2: Managing Difficulties (5 min)

Partners discuss the major difficulties they have faced to this point and the strategies they used to overcome them.

Exercise 3: Students implement their program (38 min)

At the end of class:

- Students complete a daily progress report.
- Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content. (2 min)

Session 6: Program Development Day 3

Exercise 1: Examining the Rubric (5 min)

Tell students: Before starting our research we will examine the rubric that readers will use when scoring your computational artifact and written response.

Exercise 2: Students implement their program (43 min)

At the end of class:

- Students complete a daily progress report.
- Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content. (2 min)

Session 7: Program Development Day 4

Exercise: Students implement their program (48 min)

At the end of class:

- Students complete a daily progress report.
- Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content. (2 min)

Session 8: Second Checkpoint

Tell students: Today we will evaluate program development progress and revise a program development schedule as needed. We will also once again discuss with partners the difficulties we encountered and the strategies we used to address them.

Note: Partners collaborating in their program development in the second session stages will switch to individual development at this point.

Exercise 1: Review Program Development Progress (5 min)

Each individual submits a revised schedule of program development.

Exercise 2: Managing Difficulties (5 min)

Partners discuss the major difficulties they have faced to this point and the strategies they used to overcome them.

Exercise 3: Students implement their program (38 min)

At the end of class:

- Students complete a daily progress report.
- Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content. (2 min)

Session 9: Program Development Day 5

Tell students: Today will be our next to last program development day.

Exercise: Students implement their program (48 min)

At the end of class:

- Students complete a daily progress report.
- Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content. (2 min)

Session 10: Program Development Day 6

Tell students: Today will be our last program development day.

Exercise: Students implement their program (48 min)

At the end of class:

- Students complete a daily progress report.
- Teachers will collect the progress report in order to monitor progress but may not provide any feedback to students on content. (2 min)

Session 11: Video Development Day 1

This session is the first of two scheduled to develop the video. It's possible to create the video in a single day, so if students need more time to work on their program implementation, it may be done here with video planning assigned as homework.

Tell students: Today will be our first of two days dedicated to producing the CPT video.

Exercise 1: Select Program Features (3 min)

Each individual selects features that demonstrate the programs overall purpose.

Exercise 2: Storyboard (10 min)

Students plan how to show the purpose of the program and the running of related features.

Exercise 3: Record clips for video (25 min)

Students record required clips.

Exercise 4: Video Clip Assessment (10 min)

Collaborative partners assess the video clips and recommend any additional clips or changes to existing clips.

Session 12: Video Development Day 2

Exercise 1: Video Completion (25 min)

Students complete the CPT video in mp4, wmv, avi or mov format.

Exercise 2: Draft Written Response 2a (25 min)

Students write the explanation of the programming language used, their program's purpose, and what the video illustrates.

Session 13: Written Responses: Program Development

Exercise: Students write responses to prompts 2b and 2c (50 min)

Session 14: Written Responses 2d and 2e

Exercise: Students write responses to prompts 2d and 2e (50 min)

Session 15: CPT Submission

Objectives:

- Submit all CPT documents by uploading them to the performance task submission web site.
- Present all computational innovations to the class.
- Complete a post task reflection.

Tell students: There are three goals for this session.

- Submit all three CPT documents by uploading them to the performance task submission web site.
- Present all programs to the class.
- Complete a post task reflection.

Exercise 1: Upload CPT documents (15 min)

Students upload both files.

Exercise 2: Create Program Presentations (15 min)

Students set up workstations to show their video and to give access to their program, Students perform a gallery walk to review the programs.

Exercise 3: Post Task Reflection (15 min)

Students respond to these two questions.

- What did you learn about application development?
- What advice would you give to students in next year's class?

Note: Teachers may want to record students making these comments to be used to introduce the project next year.

Evidence of Learning

Formative Assessment

Students may not receive assistance regarding the content of their CPT documents until after they have been submitted to the College Board.

Summative Assessment

Students may not receive assistance regarding the content of their CPT documents until after they have been submitted to the College Board. Once the documents have been submitted teachers should use the CPT rubric to assess performance task documents.

To convert the rubric score to a percentage teachers may want to use the following scale:

Program Development Progress: 50 pts

Over 10 days, students submit a project assessment, a program development plan and 6 daily progress reports and complete 2 program checkpoints. Weighted as 5 points for each day:

0 points no progress

1 point sporadic or minimal progress

2 points inconsistent progress

5 points consistent progress

Video Progress: 10 pts

- 5 points for each day
- 0 points no progress
- 1 point sporadic or minimal progress
- 2 points inconsistent progress
- 5 points consistent progress

Written Report Progress: 10 pts

- 5 points for each day
- 0 points no progress
- 1 point sporadic progress
- 2 points inconsistent progress
- 5 points consistent progress

Rubric score 30 pts

(http://www.umbc.edu/)


(http://www.umd.edu/)


(http://www.nsf.gov/)

Authored by: CS Matters in Maryland
Website: csmatters.org (http://csmatters.org)
Email: csmattersinmaryland@gmail.com (mailto:csmattersinmaryland@gmail.com)

Attestation and Post PD Deliverables for AP Endorsement of Computer Science Principles, FY2017
CS Matters in Maryland

Attestation represents only one component of the endorsement suite. The College Board does NOT endorse individual components for their Endorsed Providers.

Trainers

XCS Matters in Maryland attests that all trainers must review the AP Course Overview Module prior to delivering any training

X CS Matters in Maryland attests to deliver the following information to the College Board:

- A written description of how trainers are selected and trained
- A list of trainer names and qualifications
- Trainer pay rates
- An indication of how many and which trainers meet the following criteria:
  - If high school teachers, teaching for at least three years in a face-to-face classroom setting
  - If college faculty, have taught at least one college course comparable to the AP course within the past three years

Learning Event Delivery and Evaluation

XCS Matters in Maryland must provide to the College Board their own internal evaluations of their event delivery and complete a survey, sent by the College Board, that will evaluate their event delivery.

☐Yes XNo Professional development is mandatory for teachers using CS Matters in Maryland curriculum.

Provision of Data

X CS Matters in Maryland attests to deliver the following information to the College Board, by October 31 after summer training:

- Dates of training events
- Average daily attendance for each event and rate of teacher persistence in each
- Names of trainers associated with each event
- Names and schools of participant teachers
- Any feedback provided by the teachers on the training events

Day 1: Monday

Time	Title	Activities	CSP Content	Resources Daily Folder	Big Ideas and Enduring Understanding	Computational Thinking Practices
8:00-9:00	Welcome and Introduction	Revisit May content, teacher intros, KWL. More detail about the schedule Piazza invitation	Introductions	Participant list. Remember journaling	Review the 7 Big ideas. Creativity, Abstraction Data and Information, Algorithms, Programming, The Internet, Global Impact	
9:15-11:30	Getting Started Right	Give overview of curriculum and "big picture" pacing Discuss starting off the year, possible summer assignments, preparing lab/resources; the idea of gamification Model Lesson 0-3 (teachers become the students) Pair teachers to discuss prep of either 0-1 or 0-2 (one at each table); have table pairs summarize to each other	Unit 0	Resources list http://www.instituteofplay.org/work/projects/print-play-games-2/socratic-smackdown/	Enduring Understandings are built through Learning Objectives and Essential Knowledge EU 7.3 (Computing has global effects — both beneficial and harmful — on people and society.)	Practice: Connecting computing Getting students to make connections between computing innovations and their impacts.
11:30-12:30		Lunch				
12:30-1:45 2:00-3:00	Introduction to Python	Ensure all laptops are set up correctly Python instruction / building from e-book During Break: Parking Lot - What do you need to know?	What is Python, exactly? Why Python? Using Python Drawing with turtles Drawing squares, and other shapes	Python Fundamentals https://opentechschool.github.io/python-beginners/en/index.html	EU 5.4 (Programs are developed, maintained, and used by people for different purposes.)	Practice: analyzing problems and artifacts Encouraging students to plan, build, test and reflect
3:15-4:30	CSP Performance Tasks and Curriculum	Explore PT description, examples Review structure of curriculum (1-8, 3-9, Explore PT) Submission process / IRB/ forms / website	Explore Performance Task	how to get kids motivated, organized, stay current with current events, ACM	EU1.2: (Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem.)	Practice: creating artifacts & communication

7.5 **Total face-to-face training hours**

Note: This is NOT a College Board sponsored AP workshop. It is independently run by CS Matters in Maryland.

Day 2: Tuesday						
Time	**Title**	**Activities**	**CSP Content**	**Resources** Daily Folder	**Big Ideas and Enduring Understanding**	**Computational Thinking Practices**
8:00-10:00	Unit 1 Lesson Preparation	Introduction to TLO (Teacher - Learner - Observer) Unit 1 Lesson Preparation (1-3, 1-5, 1-6 (3 groups))	Unit 1: Planning to teach Innovation in Your Virtual World	Lesson prep checklist	EU 7.2 (Computing enables innovation in nearly every field.)	Practice: Connecting computing
10:15-11:30	Python Lab	Python Lab/Pair teachers based on programming experience with Python	Variables, Loops, Comments, More Efficient Squares	Values,Variables, Expressions & Iteration	EU 5.2 (People write programs to execute algorithms.)	Practice: how to foster collaboration
11:30-12:30		Lunch				
12:30-2:45	Unit 1 Teach Out	"Teach Out" with time for reflection and sharing ideas & needs.	Unit 1: Teach lessons using TLO		EU 7.4 (Computing innovations influence and are influenced by the economic, social, and cultural contexts in which they are designed and used.)	analyzing problems and artifacts
2:45-3:00		Break				
3:00-4:30	Python: Conditionals and Functions	Python based on teacher eBook	Functions, Conditionals, Logical Expressions	https://opentechschool.github.io/python-beginners/en/index.html Python Control Structures and Functional Abstraction	EU 5.5 (Programming uses mathematical and logical concepts.)	

7.5 **Total face-to-face training hours**

Note: This is NOT a College Board sponsored AP workshop. It is independently run by CS Matters in Maryland.

Day 3: Wednesday						
Time	**Title**	**Activities**	**CSP Content**	**Resources** Daily Folder	**Big Ideas and Enduring Understanding**	**Computational Thinking Practices**
8:00 - 9:15	Python Debugging and Runestone	Python Debugging and using Runestone	Runestone Interactive CS Matters Unit 2	Formative Assessment and Debugging Python	Pedagogy: planning for diverse student needs and abilities	
9:30 - 11:30	Teaching Methods Workshop	Teaching Workshop: Classroom Management, Journaling, Work in teams on differentiation for different ability levels. Discuss pacing guidelines. How to customize to your classroom	Classroom process	Suggested year-schedule timing		
11:30-12:30		Lunch			EU 2.1 (A variety of abstractions built on binary sequences can be used to represent all digital data.)	
12:30-1:30	Bits and Binary: Teacher Mastery	Bits/Binary/Hex (Lesson 1-4): Teacher Mastery	Binary and Number Systems			
1:30-1:45	Break				EU 6.1 (The Internet is a network of autonomous systems.)	
1:45- 2:45	Unit 3 Lesson Preparation	Prepare (lessons 3-1,2,3, 6, 7) using jigsaw technique	Unit 3: The Internet preparation			
3:00-4:30	Unit 3 Teach Out	Teach jigsaw	Unit 3: The Internet preparation		Pedagogy: planning for diverse student needs and abilities	

7.5 **Total face-to-face training hours**

Note: This is NOT a College Board sponsored AP workshop. It is independently run by CS Matters in Maryland.

Day 4: Thursday						
Time	**Title**	**Activities**	**CSP Content**	**Resources** Daily Folder	**Big Ideas and Enduring Understanding**	**Computational Thinking Practices**
8:00-10:00	Cryptography	Demonstration of teaching Lesson 3-10, Review crypto material (3-11,12) Introduction to www.khanacademy.org	Unit 3 Cryptography		EU 6.3 (Cybersecurity is an important concern for the Internet and the systems built on it.)	
10:15-11:30	Python: Lists and Strings	Ready to teach Python through Strings	Runestone Interactive CS Matters Unit 2 Lists and Strings	Data Abstraction: Strings and Lists	EU 2.2 (Multiple levels of abstraction are used to write programs or create other computational artifacts.)	
10:15-11:30	Advanced Python (Parallel Session)	Advanced Python: Using Python to Solve Programming Challenges	PARALLEL SESSION: Advanced Python		EU 5.2 (People write programs to execute algorithms.)	
11:30-12:30		Lunch				
12:30-2:30	Explore Performance Task: Sample Grading	Grade three sample tasks using CSM/CB rubric	Explore Task		EU 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.)	
2:45-4:30	Diversity and Differentiated Instruction	Diversity/Differentiated Instruction / Access 10K - Richard Ladner 3pm, skype name richard.ladner.43; discussion of classroom practices that support diverse populations	Equity and Access	Exercises & Worksheets for Access 10K (Dianne had these) Need projection (with sound) and ability to show a video.		Strategies to recruit students from underrepresented groups. Removing barriers to access. The case for open enrollment. Techniques to appeal to a broad audience.

7.5 **Total face-to-face training hours**

Note: This is NOT a College Board sponsored AP workshop. It is independently run by CS Matters in Maryland.

Day 5: Friday						
Time	**Title**	**Activities**	**CSP Content**	**Resources Daily Folder**	**Big Ideas and Enduring Understanding**	**Computational Thinking Practices**
8:00-9:30	Python: Practice Coding Session	Unit 2- Practice Coding	Runestone Interactive Labs Part 1	Lab Day Session 1: Variables	EU1.1: (Creative development can be an essential process for creating computational artifacts)	
9:30-9:45	botnets with Lori					
10:00-11:30	Python Teaching Methods	Unit 2 - Teaching conditions/Iteration & Active Learning	Runestone Interactive Labs Part 2	Session 2: Control Structures	EU1.3 (Computing can extend traditional forms of human expression and experience.)	
11:30-12:30	Lunch	Lunch				
12:30-2:00	Unit 3 Lesson Review and Analysis	Small groups each take a lesson, discuss, identify (1) teacher challenges and (2) student learning challenges, share back and have group discussion	3-4, 3-5, 3-14 assessment, 3-3 routing, 3-6 come up with something better than the treasure hunt.		EU 6.2 (Characteristics of the Internet influence the systems built on it.)	
2:15-4:30 (~3:45)	Unit 2 Lesson Preparation	Self-teach Unit 2 - Manipulating Strings and Lists; Prepare to teach 2-15 (day 1 and day 2) and 2-16 (day 1) strings and lists. Another group creates lesson on robot algorithms, and a fifth group works on writing pseudocode examples alligned to these three lessons (with respect to the AP Exam Reference Sheet).	Python	Session 3: Strings and Lists		

7.5 **Total face-to-face training hours**

Note: This is NOT a College Board sponsored AP workshop. It is independently run by CS Matters in Maryland.

Day 6-Monday						
Time	**Title**	**Activities**	**CSP Content**	**Resources Daily Folder**	**Big Ideas and Enduring Understanding**	**Computational Thinking Practices**
8:00-9:00	Statistics with Excel	Lesson 3-8/ Excel tutorial - still relevant? Talk about other data visualization tools	Statistics with Excel		EU 3.1 (People use computer programs to process information to gain insight and knowledge.)	
9:00-9:45	Python Lesson Preparation		Additional time to prepare to teach Python lessons			
10:00-11:30	Python Teach Out	Pilot Teachers teach 2-15 and 2-16 strings and lists	Python teaching methods			
11:30-12:30		Lunch				
12:30-1:30	Data Analysis	Unit 4 - Python file I/O, Simulation I, Basic Statistics.	Data Analysis Part I	Python File IO and Statistical Analysis DataQuest Mission	EU 3.2 (Computing facilitates exploration and the discovery of connections in information.)	
1:45-2:00	Reflection: Week Two Checkpoint	KWL, wants and needs	Week Two checkpoint			
2:00-3:00	Data Acquisition; Modeling and Simulation	Lessons 4-1 and 4-2	Acquiring and analyzing data, Modeling and Simulation		EU 2.3 (Models and simulations use abstraction to generate new understanding and knowledge.)	
3:15-4:30	Python: File I/O and Simulation	Unit 4 - File Input/Output using Python, Create a Simulation using Python	Runestone Interactive Labs Part 3	Data and Simulations in Python	EU 3.3 (There are trade-offs when representing information as digital data.)	

7.5 Total face-to-face training hours

Day 7: Tuesday						
Time	**Title**	**Activities**	**CSP Content**	**Resources** **Daily Folder**	**Big Ideas and Enduring Understanding**	**Computational Thinking Practices**
8:00-9:45	Unit 5 Lesson Prep	Unit 5 lesson prep (5-2 and 5-4) (4 groups))	Unit 5: Manipulating large data, searching and sorting			
10:00-11:30	Unit 5 Teach Out	"Teach Out" 25 minutes each with 10 minutes after each for reflection and sharing ideas & needs.	Unit 5: teach lessons			
11:30-12:30		Lunch				
12:30-1:15	Create Performance Task	Updates about the Explore Task Intro to the Create Task	Create PT introduction	PT Score Reconciliation		
1:30 - 2:45	Create Task Work Session	Teamwork discussion (20 min) Working Session (in pairs) Create Task Work Session	EarSketch or DataQuest	Create Performance Development		
3:00-4:30	Data Analysis II	--Teacher collect data(teachers use form to collect tools for the Explore task artifact) --Unit 5 - Python, DataQuest.io, do missions and parallel exercises with datasets	Data Analysis Part II	Functions, Dictionaries and List Slicing		

7.5 **Total face-to-face training hours**

Note: This is NOT a College Board sponsored AP workshop. It is independently run by CS Matters in Maryland.

Day 8: Wednesday					
Time	**Title**	**Activities**	**Resources** **Daily Folder**	**Big Ideas and Enduring Understanding**	**Computational Thinking Practices**
8:00-10:00	Advanced Algorithms	8:00 - 8:30 update concept reflection chart 8:30-9:30: Present Advanced Algorithms (5-5) 9:30-10:00:Algorithms in the Purple book (curriculum framework).	Unit 5 Advanced Algorithms	EU 4.2 (Algorithms can solve many, but not all, computational problems.)	
10:15-11:30	How to Teach Programming	Discussion and how to use Runestone LMS; how to launch Create PT to your class	Problem Solving with Algorithms and Data Structures Analyzing Algorithms	EU 4.1 (Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.)	
11:30-12:30		Lunch			
12:30-2:30	Create Performance Task Development	---Participants work on the Create task / --- small content review groups (Python basic/intermediate topics - 2 sessions 30 min max)	Create Performance Task		
2:45-4:30	Teaching Technical Writing	Teaching Technical Writing	Writing in computer classes	EU 7.1 (Computing enhances communication, interaction, and cognition.)	

7.5 **Total face-to-face training hours**

Note: This is NOT a College Board sponsored AP workshop. It is independently run by CS Matters in Maryland.

Day 9: Thursday						
Time	**Title**	**Activities**	**CSP Content**	**Resources Daily Folder**	**Big Ideas and Enduring Understanding**	**Computational Thinking Practices**
8:00-11:30	Create PT Working Session	---Create Task Working Session (10:00 break)/ --- small content review session			EU 5.1 (Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (to help people, organizations, or society).)	
11:30-12:30		Lunch				
12:30 - 2:00	Runestone LMS	Runestone Class Setup				
2:00 - 2:30	Abstraction	Managing Complexity			EU 5.3 (Programming is facilitated by appropriate abstractions.)	
2:45-4:30	Unit 3 - Internet		IPython Notebooks and Scientific Computing	Data Analysis and Visualization in Python	EU 6.1 (The Internet is a network of autonomous systems.)	

7.5 **Total face-to-face training hours**

Day 10: Friday						
Time	**Title**	**Activities**	**CSP Content**	**Resources** **Daily Folder**	**Big Ideas and Enduring Understanding**	**Computational Thinking Practices**
8:00-11:30	Create PT Working Session	8:30-11:30 Working Session (Grading)	Create Performance Task			
11:30 - 12:30		Lunch				
12:30 -1:45	Create PT: Sharing Experiences and Insights	Reconciling and Providing Feedback				
2:00-2:30	Key Takeaway	Key Takeaways	key takeaways			
2:30-3:00	Workshop Wrapup	Arrangements for community sharing of resources of curriculum tools and feedback on implementation. Structures for sharing of future resources for teaching AP Computer Science Principles. Workshop wrap up (revisit Parking Lot and KWL) & marching orders for fall	Community Building - Wrap Up	Wrap Up Session		
7		**Total face-to-face training hours**				

Note: This is NOT a College Board sponsored AP workshop. It is independently run by CS Matters in Maryland.