PISCATAWAY TOWNSHIP SCHOOLS

Dr. Frank Ranelli
Superintendent of Schools

Dr. William Baskerville
Assistant Superintendent

AP Computer Science Principles

Content Area: **Mathematics**

Grade Span: **10-12**

Revised by: **Melissa Eytchison**

Title **AP CS Teacher**

Approval Date: **August 2021**

Piscataway Township Schools
1515 Stelton Road
Piscataway, NJ 08854-1332
732 572-2289, ext. 2561
Fax 732 572-1540
www.piscatawayschools.org

COURSE OVERVIEW

Description

Full year course: 5.0 credits
Prerequisite: Algebra 2 or Introduction to Computer Science
Description: AP Computer Science Principles introduces students to the foundational concepts of computer science and challenges them to explore how computing and technology can impact the world. With a unique focus on creative problem solving and real-world application, AP Computer Science Principles prepares students for college and careers in a variety of fields. The course aligns with the 7 Big Ideas and 6 Computational Thinking Practices as laid out by the College Board for AP Computer Science Principles. Students will be given adequate in-class time to complete the two tasks that make up the AP Through-Course Assessment, and gain the essential knowledge and understanding necessary to succeed on the AP End-of-Course Exam.

Goals

AP Computer Science Principles is an introductory college-level computing course that introduces students to the breadth of the field of computer science. Students learn to design and evaluate solutions and to apply computer science to solve problems through the development of algorithms and programs. They incorporate abstraction into programs and use data to discover new knowledge. Students also explain how computing innovations and computing systems—including the internet—work, explore their potential impacts, and contribute to a computing culture that is collaborative and ethical.

Scope and Sequence

Unit	Topic	Length
Unit 1	Computational Thinking	9 Days
Unit 2	Programming	12 Days
Unit 3	Data Representation	11 Days
Unit 4	Digital Media Processing	12 Days
Unit 5	Big Data	10 Days
Unit 6	Innovative Technologies	13 Days
Unit 7	Create Performance Task	12 Days
Unit 8	Artificial Intelligence	9 Days

Resources

Platforms used:
https://www.codio.com/
https://scratch.mit.edu/
https://replit.com/

UNIT 1:

Summary and Rationale

In order to successfully master the art of creating computational artifacts, it is important that students develop a clear understanding of the complex processes and structures that make up an algorithmic solution to a given problem. In addition, it is critical that they be able to formally express those solutions clearly and unambiguously, such as what can be achieved through the use of pseudocode or a well-specified programming language. This unit focuses on introducing students to these concepts and helping them to develop the skills that they will rely on throughout the remainder of the course.

- Big Ideas covered in this unit:
- Creative Development
- Data
- Algorithms and Programming
- Impact of Computing

Recommended Pacing

9 days (1 day = 80 minutes)

Big Idea Alignment

Big Idea 2: Abstraction

- Describe the different levels of abstraction for high-level and low-level programming languages and their impact on the readability of programs. [EK 2.2.3A, EK 2.2.3B] (P3)

- Describe the hierarchical relationship between high-level and low-level programming languages in terms of programs being written by a human and executed by a computer. [EK 2.2.3C, EK 2.2.3D] (P3)

Big Idea 4: Algorithms

- Develop multiple algorithms for solving the same problem, identifying characteristics of the problem that lead to performance variations in different solutions. [EK 4.1.1H, EK 4.1.1I] (P2)

- Explain how the choice of language can improve the clarity and readability of an algorithm, but not whether an algorithmic solution exists. [EK 4.1.2F, EK 4.1.2I] (P5)

- Identify whether the number of steps required by an algorithm to solve a problem is proportional to the size of the input for the problem. [EK 4.2.1B] (P1)

- Identify problems whose solutions can be evaluated in a reasonable time. [EK 4.2.1A] (P1)

- Identify problems whose solutions cannot be evaluated in a reasonable time without the use of a heuristic. [EK 4.2.1C, EK 4.2.1D] (P1)

- Explain how heuristics are used to find quick, approximate solutions to problems that are too complex to be solved in a reasonable time, such as "find the best" or "find the smallest". [EK 4.2.2A, EK 4.2.2B, EK 4.2.2C] (P1)

- Identify problems that cannot be solved using any algorithm. [EK 4.2.2D] (P1)

- Identify problems that are undecidable and whose algorithms can produce a definitive answer for only some inputs. [EK 4.2.3A, EK 4.2.3C] (P1)

- Identify problems that are decidable and whose algorithms can produce a definitive answer for all inputs. [EK 4.2.3B] (P1)

- Analytically evaluate an algorithm's efficiency and correctness by reasoning formally or mathematically about the algorithm. [EK 4.2.4A, EK 4.2.4C] (P4)

- Evaluate different algorithms for the same problem in terms of their execution time, memory usage, and complexity. [EK 4.2.4D, EK 4.2.4E, EK 4.2.4G] (P4)

- Explain how an efficient algorithm for a problem can help solve larger instances of the problem. [EK 4.2.4F] (P4)

- Evaluate and compare the performance of linear search on any sorted or unsorted list with binary search on sorted lists. [EK 4.2.4H] (P4)

Big Idea 5: Programming

- Explain how executable programs and automation increase the scale of problems and sets of problems that can be addressed. [EK 5.2.1I, EK 5.2.1J] (P3)

- Explain how improvements in algorithms, hardware, and software increase the kinds of problems and the size of problems solvable by programming. [EK 5.2.1K] (P3)

Big Idea 6: The Internet

- Identify the security tradeoffs involved in the Internet's use of the trust model in key areas, like the domain name system (DNS) or the certificate authorities (CAs) issuing of digital certificates for validating ownership of encrypted keys used in secured communication. [EK 6.3.1A, EK 6.3.1B, EK 6.3.1M] (P1)

- Identify the software, hardware, and human components of cybersecurity. [EK 6.3.1C] (P1)

- Explain the methods and devastating effects of various forms of cyber warfare and cybercrime, including distributed denial of service attacks (DDoS), phishing, viruses, and other attacks. [EK 6.3.1D, EK 6.3.1E, EK 6.3.1F] (P1)

- Explain how antivirus software and firewalls can help prevent unauthorized access to private data. [EK 6.3.1G] (P1)

- Explain how the mathematical foundation of cryptography and the use of open standards enable the functionality and security needed for effective cybersecurity. [EK 6.3.1H, EK 6.3.1I, EK 6.3.1J] (P1)

- Explain the differences in security provided by symmetric (single-key) encryption vs. asymmetric (public key) encryption. [EK 6.3.1K, EK 6.3.1L] (P1)

Big Idea 7: Global Impact

- Explain how industries use Moore's law to plan future research and development. [EK 7.2.1F] (P1)

Instructional Focus

Unit Enduring Understandings

- CRD-1: Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.
- CRD-2: Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.
- DAT-2: Programs can be used to process data, which allows users to discover information and create new knowledge.
- AAP-2: The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.
- AAP-4: There exist problems that computers cannot solve, and even when a computer can solve a problem, it may not be able to do so in a reasonable amount of time.
- IOC-1: While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.
- IOC-2: The use of computing innovations may involve risks to your personal safety and identity.

Unit Essential Questions

- What is an algorithm?
- Is there only one approach to solving a problem?
- What is security?
- What makes a good effective team?
- What is ambiguity?
- What is the difference between high level language vs low level language?
- What is the CIA Triad?
- What is abstraction?
- What is a flow diagram?
- What is iteration, selection and sequence?
- What is Heuristics?
- What is Moore's Law?

Objectives

Students will know and be able to:

- Demonstrate effective interpersonal skills during collaboration.
- Explain how computing resources can be protected and can be misused.
- Express an algorithm that uses sequencing without using a programming language.
- Represent a step-by-step algorithmic process using sequential code statements.
- Express an algorithm that uses selection without using a programming language.
- For selection:
    - Write conditional statements
    - Determine the result of conditional statements
- Express an algorithm that uses iteration without using a programming language.
- For iteration:

    - Write iteration statements.
    - Determine the result or side-effect of iteration statements.
- Create algorithms.
- Develop a program using a development process.
- Design a program and its user interface.
- Compare multiple algorithms to determine if they yield the same side effect or result.
- For binary search algorithms:
    - Explain the requirements necessary to complete a binary search.
- For determining the efficiency of an algorithm:
    - Explain the difference between algorithms that run in reasonable time and those that do not.
    - Identify situations where a heuristic solution may be more appropriate.
- Explain the existence of undecidable problems in computer science.
- Explain how computing innovations are improved through collaboration.
- Identify the challenges associated with processing data.
- Explain how bias exists in computing innovations.
- Explain how the use of computing can raise legal and ethical concerns.

Resources

Platforms used:
https://www.codio.com/

UNIT 2:

Summary and Rationale

When used correctly, computational technologies can prove to be extremely powerful and effective tools for solving a wide range of problems. But in order to fully harness that power, an individual needs to be proficient in instructing those tools to perform highly precise operations in well-structured and logical sequences. This unit seeks to ease students into this new, structured, and more formalized way of thinking about problem-solving and programming through the use of Scratch, a block-based, visual programming language.

- Big Ideas covered in this unit:
- Creative Development
- Data
- Algorithms and Programming
- Impact of Computing

Recommended Pacing

12-days

Big Idea Alignment

Big Idea 1: Creativity

- Apply an iterative and exploratory development process to create a computational artifact using non-prescribed techniques, novel combinations of artifacts, and/or personal curiosities. [EK 1.1.1A, EK 1.1.1B] (P2)

- Design and create a computational artifact (e.g., program, image, audio, video, presentation, etc.) for creative expression using appropriate software tools and techniques (e.g., programming IDEs, spreadsheet, 3D printer, text editor, etc.). [EK 1.2.1A, EK 1.2.1B, EK 1.2.1C, EK 1.2.1D, EK 1.2.1E] (P2)

- Create a collaborative computational artifact that reflects the diverse talents and personal ideas of all group members. [EK 1.2.4E, EK 1.2.4F]

- Analyze the correctness, usability, functionality, and suitability of a computational artifact in terms of the context in which it is used or perceived. [EK 1.2.5A, EK 1.2.5C, EK 1.2.5D] (P4)

- Analyze a computational artifact for weaknesses, mistakes, and errors. [EK 1.2.5B] (P4)

Big Idea 4: Algorithms

- Express algorithms in a programming language for execution by a computer. [EK 4.1.2C] (P5)

- Construct algorithms using sequencing, selection, and iteration. [EK 4.1.2G] (P5)

Big Idea 5: Programming

- Develop a variety of programs using methods and techniques that are appropriate for the goals of the programmer. [EK 5.1.1A] (P2)

- Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge using visual, audible, or tactile inputs and outputs. [EK 5.1.1B] (P2)

- Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge using standards or methods that differ from those used for programs developed for widespread distribution. [EK 5.1.1C] (P2)

- Identify additional desired outcomes for a program that extend beyond its original purpose. [EK 5.1.1D] (P2)

- Consult and communicate with program users to identify concerns that affect the solution to problems. [EK 5.1.2G, EK 5.1.2H] (P2)

- Use effective communication between participants in the iterative development of a program. [EK 5.1.3C, EK 5.1.3F] (P6)

- Use collaboration to find and correct errors with developing programs. [EK 5.1.3D] (P6)

- Explain how algorithms are implemented using program instructions that are processed sequentially during program execution. [EK 5.2.1A, EK 5.2.1B, EK 5.2.1D] (P3)

- Explain how program instructions may involve variables that are initialized and updated, read, and written. [EK 5.2.1C] (P3)

- Explain how program execution automates processes. [EK 5.2.1E] (P3)

- Explain how one or more processes may execute on one or more CPUs, using memory, input, and output. [EK 5.2.1F, EK 5.2.1G, EK 5.2.1H] (P3)

Big Idea 7: Global Impact

- Analyze the legal and ethical concerns of open source and licensed software, libraries, and code. [EK 7.3.1F, EK 7.3.1Q] (P4)

Instructional Focus

Unit Enduring Understandings

- CRD-1: Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.
- CRD-2: Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.
- DAT-1: The way a computer represents data internally is different from the way the data are interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.
- AAP-1: To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.

- AAP-2: The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.
- AAP-3: Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.
- IOC-1: While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.

Unit Essential Questions

- Who programs?
- What is a program?
- What is block based programming?
- Why program?
- What is Scratch?
- What are variables?
- What are parameters?
- How do you change the flow of a program?
- What are custom blocks?
- What is abstraction?

Objectives

Students will know and be able to:

- Explain how computing innovations are developed by groups of people.
- Demonstrate effective interpersonal skills during collaboration.
- Explain how a program or code segment functions.
- Develop a program using a development process.
- Identify input(s) to a program.
- Represent a step-by-step algorithmic process using sequential code statements.
- Describe the purpose of a code segment or program by writing documentation.
- Design a program and its user interface.
- Explain how data can be represented using bits.
- Represent a value with a variable.
- Evaluate expressions that use arithmetic operators.
- For generating random values:
    - Write expressions to generate possible values.
    - Evaluate expressions to determine the possible results.
- For relationships between Boolean values:
    - Write expressions using logical operators.
    - Evaluate expressions that use logic operators.
- For selection:
    - Write conditional statements.
    - Determine the result of conditional statements.
- For nested selection:
    - Write nested conditional statements.

    - Determine the result of nested conditional statements.
- Compare multiple algorithms to determine if they yield the same side effect or result.
- Identify output(s) produced by a program.
- For iteration:
    - Write iteration statements.
    - Determine the result or side-effect of iteration statements.
- For procedure calls:
    - Write statements to call procedures.
    - Determine the result or effect of a procedure call.
- Describe the purpose of a code segment or program by writing documentation.
- Acknowledge code segments used from other sources.
- Explain how the use of computing can raise legal and ethical concerns.

Resources

Platforms used:
https://www.codio.com/
https://scratch.mit.edu/

UNIT 3:

Summary and Rationale

In order to make the most effective use of computational tools and data-driven applications, students need to be aware of and comfortable with the diversity of information these programs may use, as well as how that information may be digitally represented, stored, and manipulated within the computer. This unit focuses on providing students with an overview of the various levels of abstraction that are used in the digital representation of discrete data and information.

Big Ideas covered in this unit:

- Creative Development
- Data
- Algorithms and Programming
- Impact of Computing

Recommended Pacing

11 days

Big Idea Alignment

Big Idea 2: Abstraction

- Describe how digital data is represented by abstractions at different levels, including bits at the lowest level and numbers, characters, color, etc. at the higher levels. [EK 2.1.1A, EK 2.1.1B, EK 2.1.1C] (P3)

- Describe the methods and reasons for representing numbers in different bases (e.g., binary/base-2, decimal/base-10, hexadecimal/base-16, etc.) and how to convert between them. [EK 2.1.1D, EK 2.1.1E, EK 2.1.1F, EK 2.1.1G] (P3)

- Explain how finite binary sequences can be used to represent different types of data and instructions, depending on context. [EK 2.1.2A, EK 2.1.2D, EK 2.1.2E, EK 2.1.2F] (P5)

- Explain how integer representations can result in "overflow" and "underflow" errors for values that exceed the range allowed by a fixed number of bits. [EK 2.1.2B] (P5)

- Explain how real number representations can result in "round-off" errors for values that exceed the precision allowed by a fixed number of bits. [EK 2.1.2C] (P5)

- Develop an abstraction by identifying common features and removing detail in order to generalize concepts and functionality. [EK 2.2.1A, EK 2.2.1B] (P2)

- Develop an abstraction that uses parameters to enable the reuse of generalized software functionality. [EK 2.2.1C] (P2)

- Explain how binary data is processed by physical layers of computing hardware, including gates, chips, and components. [EK 2.2.3E] (P3)

- Describe the hierarchical relationship between the different levels of abstraction in computer hardware, including high-level components (e.g. video cards) and low-level components (e.g., chips, circuits, transistors, and gates). [EK 2.2.3F, EK 2.2.3G, EK 2.2.3H, EK 2.2.3I] (P3)

- Explain how applications and systems are designed, developed, and analyzed using lower-level hardware, software, and conceptual abstractions and combining them to form higher-level abstractions. [EK 2.2.3J, EK 2.2.3K] (P3)

- Use various levels of abstraction to construct a model or simulation that omits unnecessary features in order to create a simplified representation that mimics real-world events without the cost or danger of building and testing the phenomena in the real world. [EK 2.3.1A, EK 2.3.1B, EK 2.3.1C, EK 2.3.1D] (P3)

Big Idea 3: Data and Information

- Analyze how the characteristics of data, the methods and costs of manipulating the data, and the intended uses of data relate to the storage requirements and choice of storage media. [EK 3.3.1G, EK 3.3.1H, EK 3.3.1I] (P4)

Big Idea 4: Algorithms

- Express algorithms in natural language and pseudocode for human readability. [EK 4.1.2B] (P5)

Big Idea 5: Programming

- Provide documentation about program components, such as blocks and procedures, to maintain correct programs when working individually or collaboratively with other programmers. [EK 5.1.2D, EK 5.1.2E, EK 5.1.2F] (P2)

- Consult and communicate with program users to identify concerns that affect the solution to problems. [EK 5.1.2G, EK 5.1.2H] (P2)

- Use collaboration to facilitate multiple perspectives in developing ideas for solving problems by programming. [EK 5.1.3B] (P6)

- Use effective communication between participants in the iterative development of a program. [EK 5.1.3C, EK 5.1.3F] (P6)

- Employ data abstraction and its ability to separate behavior from implementation by using a variety of abstract data types, including strings, integers, floating-point numbers, and lists. [EK 5.3.1H, EK 5.3.1I, EK 5.3.1J, EK 5.3.1K] (P3)

- Use lists and procedures as abstractions in programming to produce programs that are easier to develop and maintain. [EK 5.3.1L] (P3)

- Explain how numbers and numerical concepts are expressed in programming as integers (finitely bound by storage limitations) and real numbers (approximated with limited precision). [EK 5.5.1A, EK 5.5.1B, EK 5.5.1C] (P1)

- Construct mathematical and logical expressions using arithmetic and Boolean operators. [EK 5.5.1D, EK 5.5.1E, EK 5.5.1F] (P1)

- Employ intuitive and formal reasoning about program components using Boolean concepts. [EK 5.5.1G] (P1)

- Employ lists and collections as abstract data types (ADTs) that provide functionality to add, remove, and iterate over all elements, as well as to determine whether an element is in a collection. [EK 5.5.1H, EK 5.5.1I, EK 5.5.1J] (P1)

Instructional Focus

Unit Enduring Understandings

- CRD-1: Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.
- CRD-2: Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.
- DAT-1: The way a computer represents data internally is different from the way the data are interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.
- AAP-1: To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.
- AAP-2: The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.
- IOC-1: While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.

Unit Essential Questions

- What are different ways numbers are represented?
- How do you convert between binary and decimal number systems?
- How do we digitize information?
- What is a dichotomy?
- What are Boolean expressions?
- What is abstraction?
- What are lists?
- How are lists helpful when developing abstraction?

Objectives

Students will know and be able to:

- Demonstrate effective interpersonal skills during collaboration.
- Explain how data can be represented using bits.
- For binary numbers:

    - Calculate the binary (base 2) equivalent of a positive integer (base 10) and vice versa.
    - Compare and order binary numbers.
- For binary search algorithms:
    - Determine the number of iterations required to find a value in a data set.
    - Explain the requirements necessary to complete a binary search.
- Identify inputs and corresponding expected outputs or behaviors that can be used to check the correctness of an algorithm or program.
- Evaluate expressions that use arithmetic operators.
- Explain the consequences of using bits to represent data.
- Explain how the use of computing can raise legal and ethical concerns.
- Explain how a program or code segment functions.
- Develop a program using a development process.
- Design a program and its user interface.
- Describe the purpose of a code segment or program by writing documentation.
- Acknowledge code segments used from other sources.
- Represent a step-by-step algorithmic process using sequential code statements.
- For selection:
    - Write conditional statements.
- For iteration:
    - Write iteration statements.
- Represent a list or string using a variable.
- For data abstraction:
    - Develop data abstraction using lists to store multiple elements.
    - Explain how the use of data abstraction manages complexity in program code.
- For list operations:
    - Write expressions that use list indexing and list procedures.
    - Evaluate expressions that use list indexing and list procedures.
- For algorithms involving elements of a list:
    - Write iteration statements to traverse a list.
    - Determine the result of an algorithm that includes list traversals.

Resources

Platforms used:
https://www.codio.com/
https://scratch.mit.edu/

UNIT 4:

Summary and Rationale

Building upon earlier visual programming experiences with Scratch, this unit guides students through the transition to programming in a high-level, procedural language through a brief introduction to Python. Students will become familiar with a text-based environment that more closely reflects the actual programming tools used in industry and will be better equipped for continuing studies in computer science beyond the scope of this course.

Big Ideas covered in this unit:

- Creative Development
- Data
- Algorithms and Programming
- Impact of Computing

Recommended Pacing

12 days

Big Idea Alignment

Big Idea 1: Creativity

- Create a computational artifact using computing tools and innovative, non-traditional techniques to solve a problem. [EK 1.2.2A, EK 1.2.2B] (P2)

- Create a computational artifact by combining and modifying existing artifacts to show personal expression of ideas. [EK 1.2.3A, EK 1.2.3C] (P2)

- Use computational tools to create or modify a computational artifact with enhanced detail and precision. [EK 1.2.3B] (P2)

- Use appropriate interpersonal skills, communication, and group decision-making to create an enhanced, collaborative computational artifact. [EK 1.2.4C, EK 1.2.4D] (P6)

- Create a collaborative computational artifact that reflects the diverse talents and personal ideas of all group members. [EK 1.2.4E, EK 1.2.4F] (P6)

- Identify how the creation of digital effects, images, audio, video, and animations has transformed industries. [EK 1.3.1A] (P2)

- Use computing tools to create digital audio and music by synthesizing, sampling, recording, layering, and/or looping sounds. [EK 1.3.1B] (P2)

- Use computing tools to create digital images by generating pixel patterns, manipulating digital images, or combining images. [EK 1.3.1C] (P2)

- Use appropriate software and image editing tools to create digital effects and animations. [EK 1.3.1D] (P2)

- Use computing tools to enable creative exploration of digital images and/or sounds. [EK 1.3.1E] (P2)

Big Idea 2: Abstraction

- Develop software using multiple levels of abstraction, including constants, expressions, statements, procedures, and libraries, to more effectively apply available resources and tools to solve problems. [EK 2.2.2A, EK 2.2.2B] (P3)

Big Idea 3: Data and Information

- Analyze the different trade-offs between lossy and lossless compression techniques for storing and transmitting data. [EK 3.3.1C, EK 3.3.1D, EK 3.3.1E] (P4)

Big Idea 4: Algorithms

- Develop an algorithm using sequencing, selection, and iteration. [EK 4.1.1A, EK 4.1.1B, EK 4.1.1C, EK 4.1.1D] (P2)

- Develop an algorithm that uses or combines existing, standard algorithms to ensure correctness of the resulting solution. [EK 4.1.1E, EK 4.1.1F, EK 4.1.1G] (P2)

- Explain how natural language, pseudocode, and visual and textual programming languages can all be used to express an algorithm. [EK 4.1.2A, EK 4.1.2H] (P5)

- Explain how different languages are better suited than others for expressing algorithms in specific problem domains. [EK 4.1.2D, EK 4.1.2E] (P5)

Big Idea 5: Programming

- Identify ways that advances in computing have enabled creativity in other fields. [EK 5.1.1F] (P2)

- Develop a large, correct program using an iterative process that incrementally combines tested program components. [EK 5.1.2A, EK 5.1.2B, EK 5.1.2C] (P2)

- Provide documentation about program components, such as blocks and procedures, to maintain correct programs when working individually or collaboratively with other programmers. [EK 5.1.2D, EK 5.1.2E, EK 5.1.2F] (P2)

- Develop a program using appropriate knowledge of and skill in the development process, including designing, implementing, testing, debugging, and maintaining programs. [EK 5.1.2I, EK 5.1.2J] (P2)

- Use collaboration to facilitate multiple perspectives in developing ideas for solving problems by programming. [EK 5.1.3B] (P6)

- Use abstraction to create named, parameterized, and reusable blocks of programming in order to reduce the complexity of writing and maintaining a program. [EK 5.3.1A, EK 5.3.1B, EK 5.3.1C, EK 5.3.1D] (P3)

- Use parameterization to generalize specific solutions and allow a single function to be used in place of duplicated code. [EK 5.3.1E, EK 5.3.1F, EK 5.3.1G] (P3)

- Use well-documented application program interfaces (APIs) and libraries to connect software components and to simplify complex programming. [EK 5.3.1M, EK 5.3.1N, EK 5.3.1O] (P3)

- Use good programming style, such as meaningful names for variables and procedures, shorter code blocks, and non-duplicated code, in order to improve the determination of program correctness. [EK 5.4.1A, EK 5.4.1B, EK 5.4.1C, EK 5.4.1D] (P4)

- Debug a program by locating and correcting errors. [EK 5.4.1E] (P4)

- Describe the functionality of a program at a high level in terms of what it does and how a user interacts with it and provide examples of intended behavior on specific inputs in order to find program errors. [EK 5.4.1F, EK 5.4.1G, EK 5.4.1L, EK 5.4.1M, EK 5.4.1N] (P4)

- Use visual displays (or different modalities) of program state to help in finding errors. [EK 5.4.1H] (P4)

- Justify a program's correctness by explaining how it meets its specifications. [EK 5.4.1I, EK 5.4.1J] (P4)

- Demonstrate the correctness of a program by demonstrating correctness of its components, including code blocks and procedures. [EK 5.4.1K] (P4)

Big Idea 7: Global Impact

- Analyze the legal and ethical concerns raised by innovations, access, and censorship of digital content. [EK 7.3.1A, EK 7.3.1B, EK 7.3.1C, EK 7.3.1D, EK 7.3.1E] (P4)

- Analyze the intellectual property and copyright concerns with digital information, audio, video, and textual content. [EK 7.3.1N, EK 7.3.1O, EK 7.3.1P] (P4)

Instructional Focus

Unit Enduring Understandings

- CRD-1: Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.
- CRD-2: Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.
- DAT-1: The way a computer represents data internally is different from the way the data are interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.
- AAP-1: To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.
- AAP-2: The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.

- AAP-3: Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.
- IOC-1: While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.

Unit Essential Questions

- What is Python?
- What are the benefits of block based programming?
- What is text based programming?
- What are the benefits of text based programming?
- How is media digitized?
- What is a pixel?
- What is a raster image?
- How is sound digitized?
- What are the ethical concerns of digitally manipulating photos?

Objectives

Students will know and be able to:

- Demonstrate effective interpersonal skills during collaboration.
- For errors in an algorithm or program:
    - Identify the error.
    - Correct the error.
- Represent a value with a variable.
- Determine the value of a variable as a result of an assignment.
- For data abstraction:
    - Develop data abstraction using lists to store multiple elements.
    - Explain how the use of data abstraction manages complexity in program code.
- Evaluate expressions that manipulate strings.
- For relationships between two variables, expressions, or values:
    - Write expressions using relational operators.
    - Evaluate expressions that use relational operators.
- For procedure calls:
    - Write statements to call procedures.
    - Determine the result or effect of a procedure call.
- Represent a list or string using a variable.
- For selection:
    - Write conditional statements.
    - Determine the result of conditional statements.
- For iteration:
    - Write iteration statements.
    - Determine the result or side-effect of iteration statements.
- For algorithms:
    - Create algorithms.
    - Combine and modify existing algorithms.
- For algorithms involving elements of a list:

    - a. Write iteration statements to traverse a list.
    - b. Determine the result of an algorithm that includes list traversals.
- Represent a step-by-step algorithmic process using sequential code statements.
- Explain how the use of procedural abstraction manages complexity in a program.
- Develop procedural abstractions to manage complexity in a program by writing procedures.
- Select appropriate libraries or existing code segments to use in creating new programs.
- Design a program and its user interface.
- Evaluate expressions that use arithmetic operators.
- Explain how data can be represented using bits.
- For algorithms involving elements of a list:
    - Write iteration statements to traverse a list.
- Compare data compression algorithms to determine which is best in a particular context.
- Explain how the use of computing can raise legal and ethical concerns.

Resources

Platforms used:
https://www.codio.com/
https://replit.com/

Unit 5:

Summary and Rationale

One of the most powerful applications of computational thinking relates to the creation and analysis of large data sets. In this unit, students will explore the complete set of processes and techniques that are involved in collecting large volumes of raw data and extracting new and useful information. Students will look at a variety of ways that data scientists use techniques such as data collection and extraction, data mining and analysis, and data visualization to construct and visualize new knowledge.

Big Ideas covered in this unit:

- Creative Development
- Data
- Algorithms and Programming
- Impact of Computing

Recommended Pacing

10 days

Big Idea Alignment

Big Idea 1: Creativity

- Use appropriate collaboration tools and techniques to create a computational artifact. [EK 1.2.4A, EK 1.2.4B] (P6)

Big Idea 2: Abstraction

- Use models and simulations to form and refine hypotheses and generate new knowledge about the objects or phenomena being modeled. [EK 2.3.2A, EK 2.3.2B, EK 2.3.2C, EK 2.3.2D] (P3)
- Use simulations to test hypotheses without the constraints of the real world. [EK 2.3.2E] (P3)
- Use extensive and rapid testing of models to accurately reflect the objects or phenomena being modeled. [EK 2.3.2F, EK 2.3.2H] (P3)
- Design simulations that are appropriate for the time and resource constraints of the phenomena being modeled. [EK 2.3.2G] (P3)

Big Idea 3: Data and Information

- Use computers in an iterative and interactive way to process digital information and gain insight and knowledge. [EK 3.1.1A] (P4)
- Use computational processes to filter and clean up digital information. [EK 3.1.1B] (P4)
- Use computers to process information through the combining of data sources and the clustering and classification of data. [EK 3.1.1C] (P4)

- Use computational tools to translate and transform digitally represented information to reveal patterns within the data. [EK 3.1.1D, EK 3.1.1E] (P4)

- Use collaboration to share multiple perspectives, experiences, and skill sets to generate greater insight and knowledge than can be obtained when working alone. [EK 3.1.2A, EK 3.1.2B, EK 3.1.2F] (P6)

- Use face-to-face and online collaborative tools on data-driven problems to facilitate processing information and generating greater insight and knowledge. [EK 3.1.2C, EK 3.1.2E] (P6)

- Use collaboration to develop and test hypotheses and answer questions in order to gain greater insight and knowledge. [EK 3.1.2D] (P6)

- Use appropriate visualization tools and software to communicate information about data via tables, diagrams, and textual displays. [EK 3.1.3A, EK 3.1.3B] (P5)

- Use summarization, transformation of information, and interactivity to communicate insight and knowledge gained from data. [EK 3.1.3C, EK 3.1.3D, EK 3.1.3E] (P5)

- Identify the challenges for extracting information and the opportunities for identifying trends from large data sets. [EK 3.2.1A, EK 3.2.1B] (P1)

- Use appropriate search and filtering tools to efficiently find and make connections with information in large data sets. [EK 3.2.1C, EK 3.2.1D, EK 3.2.1E] (P1)

- Use appropriate software tools, such as spreadsheets and databases, to efficiently organize and find trends in information. [EK 3.2.1F] (P1)

- Use metadata to add descriptive information about the organization or contents of an image, a Web page, or other complex objects in order to increase the searchability or usefulness of the data. [EK 3.2.1G, EK 3.2.1H, EK 3.2.1I] (P1)

- Use large data sets to store, retrieve, and computationally process information such as transactions, measurements, text, sound, images, and video. [EK 3.2.2A] (P3)

- Identify the challenges of structuring, storing, processing and curating large data sets. [EK 3.2.2B, EK 3.2.2C] (P3)

- Identify the challenges of maintaining privacy with large data sets that contain personal information. [EK 3.2.2D] (P3)

- Analyze the way that the size of a data set affects its scalability and the computational and analytical techniques required to effectively store, manage, transmit, and process data. [EK 3.2.2E, EK 3.2.2F, EK 3.2.2G, EK 3.2.2H] (P3)

- Analyze how security and privacy concerns involve trade-offs and impact the methods of storing and transmitting information. [EK 3.3.1A, EK 3.3.1B, EK 3.3.1F] (P4)

Big Idea 4: Algorithms

- Empirically evaluate an algorithm by implementing the algorithm and running it on different inputs. [EK 4.2.4B] (P4)

Big Idea 5: Programming

- Explain how 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. [EK 5.1.1E] (P2)

- Use collaboration to decrease the size and complexity of tasks required of individual programmers and to develop program components independently. [EK 5.1.3A, EK 5.1.3E] (P6)

Big Idea 7: Global Impact

- Explain how the advantages of distributed computing and crowdsourcing affect the ability to solve large-scale problems related to digital data (e.g., "citizen science", SETI@Home, Amazon's Mechanical Turk, etc.). [EK 7.1.2A, EK 7.1.2B, EK 7.1.2C, EK 7.1.2D, EK 7.1.2E, EK 7.1.2F] (P4)

- Explain how new technologies and applications are made possible by the proliferation of always-on mobile computers. [EK 7.1.2G] (P4)

- Explain how machine learning and data mining have enabled innovations in medicine, business, and science. [EK 7.2.1A] (P1)

- Explain how computing enables innovation and creativity in scientific and other fields. [EK 7.2.1B, EK 7.2.1C, EK 7.2.1G] (P1)

- Explain how open access to digital information and scientific databases have benefited scientific researchers. [EK 7.2.1D, EK 7.2.1E] (P1)

- Analyze the privacy and security concerns related to the collection, aggregation, and use of personal data. [EK 7.3.1G, EK 7.3.1H, EK 7.3.1I, EK 7.3.1J, EK 7.3.1K, EK 7.3.1L, EK 7.3.1M] (P4)

- Use online databases and libraries to access information. [EK 7.5.1A] (P1)

- Use advanced search tools, Boolean logic, and key words to focus or limit searches to desired results. [EK 7.5.1B] (P1)

- Avoid plagiarism by appropriately acknowledging sources. [EK 7.5.1C] (P1)

- Evaluate the credibility and relevance of sources of information. [EK 7.5.2A, EK 7.5.2B] (P5)

Instructional Focus

Unit Enduring Understandings

- CRD-1: Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.
- DAT-2: Programs can be used to process data, which allows users to discover information and create new knowledge.

- AAP-3: Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.
- IOC-1: While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.
- IOC-2: The use of computing innovations may involve risks to your personal safety and identity.

Unit Essential Questions

- What is Big Data?
- What information can be extracted from data?
- What is privacy vs utility?
- What are TOS?
- What are the ethical concerns around data privacy.
- How can data be stored, transformed and secured?
- What is Metadata?

Objectives

Students will know and be able to:

- Demonstrate effective interpersonal skills during collaboration.
- Describe what information can be extracted from data.
- Explain how the use of computing can raise legal and ethical concerns.
- Explain how programs can be used to gain insight and knowledge from data.
- Identify the challenges associated with processing data.
- Extract information from data using a program.
- Describe the risks to privacy from collecting and storing personal data on a computer system.
- For simulations:
    - Explain how computers can be used to represent real-world phenomena or outcomes.
    - Compare simulations with real-world contexts.
- Explain how people participate in problem-solving processes at scale.

Resources

Platforms used:
https://www.codio.com/
https://replit.com/

Unit 6:

Summary and Rationale

As a way of further expanding upon the applications of computer science in the advancement of computational technologies, this unit aims to broaden students' awareness of the computing tools they use

and rely on every day and to encourage them to start thinking about the decisions and processes that go into the creation of these technologies.

Big Ideas covered in this unit:

- Creative Development
- Data
- Algorithms and Programming
- Computer Systems and Networks
- Impact of Computing

Recommended Pacing

13 days

Big Idea Alignment

Big Idea 1: Creativity

- Apply an iterative and exploratory development process to create a computational artifact using non-prescribed techniques, novel combinations of artifacts, and/or personal curiosities. [EK 1.1.1A, EK 1.1.1B] (P2)

- Use appropriate collaboration tools and techniques to create a computational artifact. [EK 1.2.4A, EK 1.2.4B] (P6)

- Use appropriate interpersonal skills, communication, and group decision-making to create an enhanced, collaborative computational artifact. [EK 1.2.4C, EK 1.2.4D] (P6)

- Create a collaborative computational artifact that reflects the diverse talents and personal ideas of all group members. [EK 1.2.4E, EK 1.2.4F] (P6)

Big Idea 5: Programming

- Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge using visual, audible, or tactile inputs and outputs. [EK 5.1.1B] (P2)

- Identify additional desired outcomes for a program that extend beyond the original purpose of a program. [EK 5.1.1D] (P2)

- Explain how 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. [EK 5.1.1E] (P2)

- Use collaboration to facilitate multiple perspectives in developing ideas for solving problems by programming. [EK 5.1.3B] (P6)

- Use effective communication between participants in the iterative development of a program. [EK 5.1.3C, EK 5.1.3F] (P6)

Big Idea 6: The Internet

- Explain how world-wide collaboration is enabled through the end-to-end architecture that consists of unique addresses and standard protocols for connecting new devices and networks on the Internet. [EK 6.1.1A, EK 6.1.1B, EK 6.1.1C, EK 6.1.1D] (P3)

- Explain how the domain name system (DNS) translates names to IP addresses that are assigned to every device connected to the Internet. [EK 6.1.1E, EK 6.1.1G] (P3)

- Explain the role of evolving Internet standards and its relation to the need for a new Internet protocol (IPv6). [EK 6.1.1F, EK 6.1.1H] (P3)

- Explain how the Internet Engineering Task Force (IETF) establishes and oversees key Internet standards, such as hypertext transfer protocol (HTTP), Internet protocol (IP), and simple mail transfer protocol (SMTP). [EK 6.1.1I] (P3)

- Explain how the hierarchical design of the Internet's routing and addressing systems (domain name syntax, IP addresses) provide fault tolerance and redundancy. [EK 6.2.1A, EK 6.2.1B, EK 6.2.1C, EK 6.2.1D] (P5)

- Explain how the hierarchy and redundancy of routing with the domain name system (DNS) help the Internet to scale to more devices and more people. [EK 6.2.2A, EK 6.2.2B, EK 6.2.2C] (P4)

- Explain how open standards and well-specified interfaces and protocols enable widespread growth and use of the Internet. [EK 6.2.2D, EK 6.2.2E] (P4)

- Explain the importance of standards in sharing and transmitting data and control information through a packet-switched system, such as transmission control protocol/Internet protocol (TCP/IP), hypertext transfer protocol (HTTP), and secure sockets layer/transport layer security (SSL.TLS). [EK 6.2.2F, EK 6.2.2G, EK 6.2.2H] (P4)

- Explain how the bandwidth and latency of a system affect its use. [EK 6.2.2I, EK 6.2.2J, EK 6.2.2K] (P4)

Big Idea 7: Global Impact

- Analyze the impact and use of today's communication technologies on their lives (including email, SMS, chat, video conferencing, social media, etc.). [EK 7.1.1A, EK 7.1.1B, EK 7.1.1C, EK 7.1.1H] (P4)

- Analyze the advantages and solutions enabled by cloud computing and instant access to public data (e.g., search engines, wikis, product reviews, etc.). [EK 7.1.1D, EK 7.1.1E, EK 7.1.1F] (P4)

- Identify new technologies and applications made possible by the proliferation of inexpensive sensors and processors. [EK 7.1.1G] (P4)

- Analyze how sensor-driven technology (e.g., GPS, sensor networks, smart grids/building/transportation, assistive technologies, etc.) have changed human behavior and enhanced human capabilities. [EK 7.1.1I, EK 7.1.1J, EK 7.1.1K, EK 7.1.1L] (P4)

- Analyze the impact and use of the Internet and the Web for communication, e-commerce, healthcare, entertainment, and online learning. [EK 7.1.1M, EK 7.1.1N] (P4)

- Identify the positive and negative effects of the Internet and the Web on productivity. [EK 7.1.1O] (P4)

- Explain how social media, online access, and the "digital divide" affect individuals and socioeconomic groups differently around the world. [EK 7.4.1A, EK 7.4.1D] (P1)

- Explain how mobile, wireless, and networked computing impact innovation throughout the world. [EK 7.4.1B] (P1)

- Explain how the global distribution of computing resources raises issues of equity, access, and power. [EK 7.4.1C] (P1)

- Explain commercial and governmental initiatives support networks and infrastructure. [EK 7.4.1E] (P1)

Instructional Focus

Unit Enduring Understandings

- CRD-1: Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.
- CRD-2: Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.
- DAT-2: Programs can be used to process data, which allows users to discover information and create new knowledge.
- AAP-1: To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.
- CSN-1: Computer systems and networks facilitate the transfer of data.
- CSN-2: Parallel and distributed computing leverage multiple computers to more quickly solve complex problems or process large data sets.
- IOC-1: While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.
- IOC-2: The use of computing innovations may involve risks to your personal safety and identity.

Unit Essential Questions

- What is a computing innovation?
- How can the effect of a computing innovation be both beneficial and harmful?
- What is a Network?
- How does a Network work?
- What is the Internet of Things?
- How does the internet work?
- Is the internet and the World Wide Web the same?
- WHat is the Digital Divide?
- What is Net Neutrality?
- What is the difference between sequential, parallel, and distributed computing?

Objectives

Students will know and be able to:

- Demonstrate effective interpersonal skills during collaboration.
- Explain how computing innovations are improved through collaboration.
- Explain how computing innovations are developed by groups of people.
- Describe the purpose of a computing innovation.
- Identify input(s) to a program.
- Identify output(s) produced by a program.
- Represent a value with a variable.
- Explain how an effect of a computing innovation can be both beneficial and harmful.
- Explain how a computing innovation can have an impact beyond its intended purpose.
- Describe the risks to privacy from collecting and storing personal data on a computing system.
- Explain how unauthorized access to computing resources is gained.
- Describe what information can be extracted from metadata.
- Extract information from data using a program.
- Describe issues that contribute to the digital divide.
- Explain how the use of computing can raise legal and ethical concerns.
- Explain how computing devices work together in a network.
- Explain how the internet works.
- Explain how data are sent through the internet via packets.
- For fault-tolerant systems, like the internet:
    - Describe the benefits of fault tolerance.
    - Explain how a given system is fault-tolerant.
- Describe the differences between the internet and the World Wide Web.
- For sequential, parallel, and distributed computing:
    - Compare problem solutions.
    - Determine the efficiency of solutions.
- Describe benefits and challenges of parallel and distributed computing.
- Identify the challenges associated with processing data.
- Explain how computing resources can be protected and can be misused.
- Explain how unauthorized access to computing resources is gained.

Resources

Platforms used:
https://www.codio.com/

Unit 7:

Summary and Rationale

AP Computer Science Principles Performance task is a student created program that accounts for 30% of a student's AP Score. Performance Task Guidelines:
https://apcentral.collegeboard.org/pdf/ap-csp-student-task-directions.pdf

Recommended Pacing

12 days

Big Idea Alignment

Big Idea 1: Creativity

- Design and create a computational artifact (e.g., program, image, audio, video, presentation, etc.) for creative expression using appropriate software tools and techniques (e.g., programming IDEs, spreadsheet, 3D printer, text editor, etc.). [EK 1.2.1A, EK 1.2.1B, EK 1.2.1C, EK 1.2.1D, EK 1.2.1E] (P2)
- Create a computational artifact using computing tools and innovative, non-traditional techniques to solve a problem. [EK 1.2.2A, EK 1.2.2B] (P2)
- Create a computational artifact by combining and modifying existing artifacts to show personal expression of ideas. [EK 1.2.3A, EK 1.2.3C] (P2)
- Use computational tools to create or modify a computational artifact with enhanced detail and precision. [EK 1.2.3B] (P2)
- Use appropriate collaboration tools and techniques to create a computational artifact. [EK 1.2.4A, EK 1.2.4B] (P6)
- Use appropriate interpersonal skills, communication, and group decision-making to create an enhanced, collaborative computational artifact. [EK 1.2.4C, EK 1.2.4D] (P6)
- Create a collaborative computational artifact that reflects the diverse talents and personal ideas of all group members. [EK 1.2.4E, EK 1.2.4F] (P6)
- Analyze the correctness, usability, functionality, and suitability of a computational artifact in terms of the context in which it is used or perceived. [EK 1.2.5A, EK 1.2.5C, EK 1.2.5D] (P4)
- Analyze a computational artifact for weaknesses, mistakes, and errors. [EK 1.2.5B] (P4)

Big Idea 2: Abstraction

- Develop an abstraction by identifying common features and removing detail in order to generalize concepts and functionality. [EK 2.2.1A, EK 2.2.1B]
- Develop software using multiple levels of abstraction, including constants, expressions, statements, procedures, and libraries, to more effectively apply available resources and tools to solve problems. [EK 2.2.2A, EK 2.2.2B]

Big Idea 3: Data and Information

- Analyze how the characteristics of data, the methods and costs of manipulating the data, and the intended uses of data relate to the storage requirements and choice of storage media. [EK 3.3.1G, EK 3.3.1H, EK 3.3.1I] (P4)

Big Idea 4: Algorithms

- Develop an algorithm using sequencing, selection, and iteration. [EK 4.1.1A, EK 4.1.1B, EK 4.1.1C, EK 4.1.1D] (P2)

- Develop an algorithm that uses or combines existing, standard algorithms to ensure correctness of the resulting solution. [EK 4.1.1E, EK 4.1.1F, EK 4.1.1G] (P2)

- Express algorithms in natural language and pseudocode for human readability. [EK 4.1.2B]

- Express algorithms in a programming language for execution by a computer. [EK 4.1.2C]

- Construct algorithms using sequencing, selection, and iteration. [EK 4.1.2G]

Big Idea 5: Programming

- Develop a variety of programs using methods and techniques that are appropriate for the goals of the programmer. [EK 5.1.1A] (P2)

- Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge using visual, audible, or tactile inputs and outputs. [EK 5.1.1B] (P2)

- Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge using standards or methods that differ from those used for programs developed for widespread distribution. [EK 5.1.1C] (P2)

- Develop a large, correct program using an iterative process that incrementally combines tested program components. [EK 5.1.2A, EK 5.1.2B, EK 5.1.2C] (P2)

- Provide documentation about program components, such as blocks and procedures, to maintain correct programs when working individually or collaboratively with other programmers. [EK 5.1.2D, EK 5.1.2E, EK 5.1.2F] (P2)

- Consult and communicate with program users to identify concerns that affect the solution to problems. [EK 5.1.2G, EK 5.1.2H] (P2)

- Develop a program using appropriate knowledge and skill of the development process, including designing, implementing, testing, debugging, and maintaining programs. [EK 5.1.2I, EK 5.1.2J] (P2)

- Use collaboration to decrease the size and complexity of tasks required of individual programmers and to develop program components independently. [EK 5.1.3A, EK 5.1.3E] (P6)

- Use collaboration to facilitate multiple perspectives in developing ideas for solving problems by programming. [EK 5.1.3B] (P6)

- Use effective communication between participants in the iterative development of a program. [EK 5.1.3C, EK 5.1.3F] (P6)

- Use collaboration to find and correct errors with developing programs. [EK 5.1.3D] (P6)

- Explain how algorithms are implemented using program instructions that are processed sequentially during program execution. [EK 5.2.1A, EK 5.2.1B, EK 5.2.1D] (P3)

- Explain how program instructions may involve variables that are initialized and updated, read, and written. [EK 5.2.1C] (P3)

- Explain how executable programs and automation increase the scale of problems and sets of problems that can be addressed. [EK 5.2.1I, EK 5.2.1J] (P3)

- Use abstraction to create named, parameterized, and reusable blocks of programming in order to reduce the complexity of writing and maintaining a program. [EK 5.3.1A, EK 5.3.1B, EK 5.3.1C, EK 5.3.1D] (P3)

- Use parameterization to generalize specific solutions and allow a single function to be used in place of duplicated code. [EK 5.3.1E, EK 5.3.1F, EK 5.3.1G] (P3)

- Employ data abstraction and its ability to separate behavior from implementation by using a variety of abstract data types, including strings, integers, floating-point numbers, and lists. [EK 5.3.1H, EK 5.3.1I, EK 5.3.1J, EK 5.3.1K] (P3)

- Use lists and procedures as abstractions in programming to produce programs that are easier to develop and maintain. [EK 5.3.1L] (P3)

- Use well-documented application program interfaces (APIs) and libraries to connect software components and to simplify complex programming. [EK 5.3.1M, EK 5.3.1N, EK 5.3.1O] (P3)

- Use good programming style, such as meaningful names for variables and procedures, shorter code blocks, and non-duplicated code, in order to improve the determination of program correctness. [EK 5.4.1A, EK 5.4.1B, EK 5.4.1C, EK 5.4.1D] (P4)

- Debug a program by locating and correcting errors. [EK 5.4.1E] (P4)

- Describe the functionality of a program at a high level in terms of what it does and how a user interacts with it and provide examples of intended behavior on specific inputs in order to find program errors. [EK 5.4.1F, EK 5.4.1G, EK 5.4.1L, EK 5.4.1M, EK 5.4.1N] (P4)

- Use visual displays (or different modalities) of program state to help in finding errors. [EK 5.4.1H] (P4)

- Construct mathematical and logical expressions using arithmetic and Boolean operators. [EK 5.5.1D, EK 5.5.1E, EK 5.5.1F] (P1)

- Employ intuitive and formal reasoning about program components using Boolean concepts. [EK 5.5.1G] (P1)

- Employ lists and collections as abstract data types (ADTs) that provide functionality to add, remove, and iterate over all elements, as well as to determine whether an element is in a collection. [EK 5.5.1H, EK 5.5.1I, EK 5.5.1J] (P1)

Instructional Focus

Performance Task Guidelines: https://apcentral.collegeboard.org/pdf/ap-csp-student-task-directions.pdf
Resources
Platforms used: **https://www.codio.com/** **https://scratch.mit.edu/**

Unit 8:

Summary and Rationale
As computing devices grow more powerful, they are increasingly used to augment or replace human labor through the simulation of human skills and behaviors. In this unit, you will explore some of the ways in which computer scientists incorporate "artificial intelligence (AI)" into their algorithms, and what the philosophical implications of these advances may mean for the future. You will examine the use of a number of AI-infused applications, and finally, evaluate these applications using the standard metric for general AI—the *Turing Test*. (This to supplemental material to be completed after the AP Exam)
Recommended Pacing

9 days

Big Idea Alignment

Big Idea 2: Abstraction

- Describe the different levels of abstraction for high-level and low-level programming languages and their impact on the readability of programs. [EK 2.2.3A, EK 2.2.3B] (P3)
- Describe the hierarchical relationship between high-level and low-level programming languages in terms of programs being written by a human and executed by a computer. [EK 2.2.3C, EK 2.2.3D] (P3)

Big Idea 4: Algorithms

- Develop multiple algorithms for solving the same problem, identifying characteristics of the problem that lead to performance variations in different solutions. [EK 4.1.1H, EK 4.1.1I] (P2)
- Explain how the choice of language can improve the clarity and readability of an algorithm, but not whether an algorithmic solution exists. [EK 4.1.2F, EK 4.1.2I] (P5)
- Identify whether the number of steps required by an algorithm to solve a problem is proportional to the size of the input for the problem. [EK 4.2.1B] (P1)
- Identify problems that are decidable and whose algorithms can produce a definitive answer for all inputs. [EK 4.2.3B] (P1)
- Analytically evaluate an algorithm's efficiency and correctness by reasoning formally or mathematically about the algorithm. [EK 4.2.4A, EK 4.2.4C] (P4)
- Evaluate different algorithms for the same problem in terms of their execution time, memory usage, and complexity. [EK 4.2.4D, EK 4.2.4E, EK 4.2.4G] (P4)

Instructional Focus

Unit Enduring Understandings

- CRD-1: Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.
- CRD-2: Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.
- DAT-2: Programs can be used to process data, which allows users to discover information and create new knowledge.
- AAP-1: To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.

Unit Essential Questions

- Who is Alan Turning?
- What is the Turing Test?
- What is Blackbox Testing?
- What is intelligence?

- What is language ambiguity?

Objectives

Students will know and be able to::

- Explain how computing innovations are developed by groups of people.
- Demonstrate effective interpersonal skills during collaboration.
- Explain how a program or code segment functions.
- Develop a program using a development process.
- Identify input(s) to a program.
- Represent a step-by-step algorithmic process using sequential code statements.
- Describe the purpose of a code segment or program by writing documentation.
- Design a chatbot and its user interface.
- Represent a value with a variable.
- For relationships between Boolean values:
    - Write expressions using logical operators.
    - Evaluate expressions that use logic operators.
- For selection:
    - Write conditional statements.
    - Determine the result of conditional statements.
- For nested selection:
    - Write nested conditional statements.
    - Determine the result of nested conditional statements.
- Identify output(s) produced by a program.
- For iteration:
    - Write iteration statements.
    - Determine the result or side-effect of iteration statements.
- For procedure calls:
    - Write statements to call procedures.
    - Determine the result or effect of a procedure call.
- Describe the purpose of a code segment or program by writing documentation.
- Acknowledge code segments used from other sources.

Resources

Platforms used:

https://www.codio.com/
https://scratch.mit.edu/