Computational Thinking Concepts Guide

Overview

In this guide you will find eleven terms and definitions for Computational Thinking (CT) concepts. These concepts can be incorporated into existing lesson plans, projects, and demonstrations in order to infuse CT into any disciplinary subject.

Concepts, Definitions, and Teaching Tips

Abstraction is identifying and extracting relevant information to define main idea(s)

Teaching Tips:
- Show how a daily planner uses abstraction to represent a week in terms of days and hours, helping us to organize our time
- Find the main idea (thesis) of a short story or article to help students understand the important information

Algorithm Design is creating an ordered series of instructions for solving similar problems or for doing a task

Teaching Tips:
- Give the example of how in mathematics, when we add and subtract fractions with different denominators, we follow an algorithm
- Give the example of when a chef writes a recipe for a dish, she is creating an algorithm that others can follow to replicate the dish

Automation is having computers or machines do repetitive tasks

Teaching Tips:
- Show how automation can be used to perform tasks that would take a very long time to complete using a manual process, such as identifying the migration patterns of a specific demographic based on census data

Data Collection is gathering information

Teaching Tips:
- Have students engage in the process of collecting data by having them gather the birthday and gender of their peers and record it in a spreadsheet

Data Analysis is making sense of data by finding patterns or developing insights

Teaching Tips:
- Ask students to formulate their own questions that can be addressed with data collection; work through data together to answer them
- Using data sets with anonymized personal data, such as height, shoe size, favorite color, etc., show how data analysis can be used to highlight information that is meaningful and relevant to your students

Data Representation is depicting and organizing data in appropriate graphs, charts, words, or images

Teaching Tips:
- Plot data manually on the whiteboard or via projector so that your students can see the process of how the organization unfolds
- Ask your students which visual representation is best for a given data set and work through some less relevant visualizations first. Discuss as a class why the data should be represented in one way or another.

Decomposition is breaking down data, processes, or problems into smaller, manageable parts

Teaching Tips:
- In mathematics, we can decompose a number such as 256.37 as follows: $2*10^2+5*10^1+6*10^0+3*10^{-1}+7*10^{-2}$
- In science we decompose a projectile's velocity into its components along the x- and y-axis

Parallelization is simultaneous processing of smaller tasks from a larger task to more efficiently reach a common goal

Teaching Tips:
- An example of parallelization in computing is when a single task (such as the analysis of a DNA sequence) is broken into smaller tasks and simultaneously analyzed by different computers so that the analysis can be processed more efficiently
- Define the common goal at the outset, and then have your students work in groups. Assign different tasks to each group to work toward the common goal.

Pattern Generalization is creating models, rules, principles, or theories of observed patterns to test predicted outcomes

Teaching Tips:
- In mathematics, we write generalized formulas in terms of variables instead of numbers so that we can use them to solve problems involving different values
  - The slope of any straight line can be described as a function of $y = mx + b$
- In science, we use theories to describe the generalized mechanism by which natural phenomena occur

Pattern Recognition is observing patterns, trends, and regularities in data

Teaching Tips:
- Have your students identify trends in stock price cycles that may suggest when they should be bought and sold

Simulation is developing a model to imitate real-world processes

Teaching Tips:
- Have your students illustrate the movement of a solar system by modeling the gravitationally curved path of an object around a point in space

---

More reference docs, lesson plans, and demonstrations can be found on Google's Exploring Computational Thinking website (g.co/exploringCT)

Operational Definition of Computational Thinking for K–12 Education

The International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) have collaborated with leaders from higher education, industry, and K–12 education to develop an operational definition of computational thinking. The operational definition provides a framework and vocabulary for computational thinking that will resonate with all K–12 educators. ISTE and CSTA gathered feedback by survey from nearly 700 computer science teachers, researchers, and practitioners who indicated overwhelming support for the operational definition.

Computational thinking (CT) is a problem-solving process that includes (but is not limited to) the following characteristics:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them.
- Logically organizing and analyzing data
- Representing data through abstractions such as models and simulations
- Automating solutions through algorithmic thinking (a series of ordered steps)
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- Generalizing and transferring this problem solving process to a wide variety of problems

These skills are supported and enhanced by a number of dispositions or attitudes that are essential dimensions of CT. These dispositions or attitudes include:

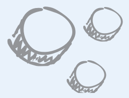
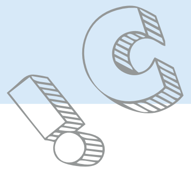
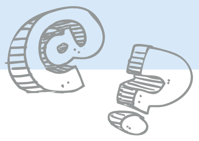
- Confidence in dealing with complexity
- Persistence in working with difficult problems
- Tolerance for ambiguity
- The ability to deal with open ended problems
- The ability to communicate and work with others to achieve a common goal or solution

csta Computer Science Teachers Association

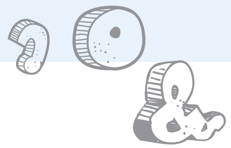
iste

CT concept or capability	Definition	What to do	Why is this important?
Data Collection	The process of gathering appropriate information	When developing solutions that a computer can execute, determine what data is needed and appropriate to use in the problem solving process.	It is necessary to use relevant data in the problem solving process. There is a saying "Garbage in, garbage out".
Data Analysis	Making sense of data, finding patterns, and drawing conclusions	When developing solutions that a computer can execute, analyze output data to understand and improve the process applied to the input data.	it is necessary to analyze output data to assess the outcome of a process and/or correctness of a solution.
Data Representation	Depicting and organizing data in appropriate graphs, charts, words, or images	When coming up with solutions that computers can execute, plan ahead for data input and output, data storage, and easy data retrieval. Come up with compact representations. Develop a plan for visualizing data sets.	Easy storage and retrieval of data will make your code faster and memory usage more efficient.
Problem Decomposition	Breaking down tasks into smaller, manageable parts	When coming up with solutions that computers can execute, break long sequences of code down into smaller parts to write more compact, reuseable code.	Smaller reuseable chunks of code (subprocedures) are easier to debug, test, and reuse. Sometimes breaking things down also enables us to attack problems as independent smaller problems.
Abstraction	Reducing complexity to define main idea	When coming up with solutions that computers can execute, develop simple representations that can stand for the set of all such parts. Reduce detail and complexity to define the main idea. Seek regularity so one can stand for all.	To achieve scale and wide applicability of solutions develop general case solutions not solutions to one instance of a problem.
Algorithms & Procedures	Series of ordered steps taken to solve a problem or achieve some end.	When coming up with solutions that computers can execute, consider the capabilities of the computer and the language specific commands it can follow. Develop a solution using a sequence of instructions that the computer can execute.	Computers only follow instructions they are given in a language they can understand.
Automation	Having computers or machines do repetitive or tedious tasks.	When coming up with solutions that computers can execute, describe how a machine or computer could be used to complete repetitive tasks.	Computers can easily perform repetitions - thus they can easily repeat a task, search a long list, produce all possible combinations, and/or do random selection.
Parallelization	Organize resources to simultaneously carry out tasks to reach a common goal.	Develop a plan on how the computer can be used to carry out more than one process at the same time.	Computers (those with multiple processors) can readily perform multiple processes at once but when doing so, it is important to make sure tasks do not interfere with one another.
Simulation	Dynamic representation or model of a process. Simulation also involves running experiments using models.	When studying and potentially solving real-world problems, develop a dynamic computer model of a real-world scenario within which to run experiments.	Computers can be used to run experiments that take place in virtual worlds. This is specially important when the scenario or process being studied is too fast, too slow, too expensive, or too dangerous to experiment with in the real-world.

C | CT Vocabulary and Progression Chart

	Definition	Grades PK to 2	Grades 3 to 5	Grades 6 to 8	Grades 9 to 12
Data Collection	The process of gathering appropriate information	Conduct an experiment to find the fastest toy car down an incline and record the order of cars across the finish line in a chart.	Review examples of writing to identify strategies for writing an essay.	Design survey questions to gather appropriate information to answer questions (e.g., asking fellow students if they were absent from school in the past month and whether they were suffering from the flu).	Students develop a survey and collect both qualitative and quantitative data to answer the question: "Has global warming changed the quality of life?"
Data Analysis	Making sense of data, finding patterns, and drawing conclusions	Make generalizations about the order of finishing a toy car race based on the characteristics of the car with a focus on weight. Test conclusions by adding weight to cars to change results.	Categorize strong and weak examples of writing samples to develop a rubric.	Produce and evaluate charts from data generated by a digital probe and describe trends, patterns, variations, and/or outliers represented in the chart.	Use appropriate statistical methods that will best test the hypothesis: "Global warming has not changed the quality of life." 
Data Representation	Depicting and organizing data in appropriate graphs, charts, words, or images	Create a chart or a line drawing that shows how the speed of a toy car changes when its weight is changed.	Match each writing sample to the rubric and create a chart showing which example best fits in each category of the rubric.	Plot data using different charting formats and select the most effective visual representation strategy.	Groups of students represent the same data in different ways based on a position relating to the question: "Has global warming changed the quality of life?" Different representations may result in varying conclusions.
Problem Decomposition	Breaking down tasks into smaller, manageable parts	Create directions to a location in the school by breaking the directions down into smaller geographical zones. Join the sections of directions together into a whole.	Develop a plan to make the school "green." Separate strategies such as recycling paper and cans, reducing use of electricity, and composting food waste.	In planning the publication of a monthly newsletter, identify roles, responsibilities, timeline, and resources needed to complete the project.	Consider the large-scale problem: "What does it take to become a rock star?" Break it into smaller parts. Discuss what variables are within a student's control and what variables are determined by outside factors.
Abstraction	Reducing complexity to define main idea 	With many sizes and colors of three-sided shapes, the abstract is a triangle. 	Hear a story, reflect on main items, and determine an appropriate title.	After studying a period in history, identify symbols, themes, events, key people, and values that are most representative of the time period (e.g., coat of arms).	Choose a period in politics that was most like the current one by analyzing the essential characteristics of the current period.

	Definition	Grades PK to 2	Grades 3 to 5	Grades 6 to 8	Grades 9 to 12
Algorithms & Procedures	Series of ordered steps taken to solve a problem or achieve some end.	Create a set of directions from the school to the major landmarks in the neighborhood.	Design a board game and write instructions to play. Test instructions on peers trying to play the game. Refine instructions with feedback from peers who played the game.	Program a robot to find its way out of a maze such that given any maze, the robot could exit successfully within a specified time period.	Discuss the decision-making process for choosing a college, then create an algorithm that describes that process. The algorithm will be able to handle unknown variables, such as where friends are attending, availabilty of financial aid, and admission success, to come to an unambiguous decision.
Automation	Having computers or machines do repetitive or tedious tasks.	Converse with a classroom in another state or country to learn about their culture using Internet-based tools to replace writing letters.	Investigate what automation is through real-world examples, like barcodes, teller machines, and library bar codes.	Program a sensor to collect pollution data (set timers with probes) and then use a computer program to sort the readings from maximum to minimum CO_2 levels.	Debate the merits of learning skills and information that are rarely necessary today because of automation. These skills might include long division, deriving square roots, spelling, statistical formulas, memorizing historic dates, etc.
Simulation	Representation or model of a process. Simulation also involves running experiments using models.	After a set of directions has been created, act out the steps to be sure they are correct.	Create an animation to demonstrate the understanding of a process.	Use a model of a simple ecosystem to conduct experiments that answer what happens to the ecosystem if some percentage of the producers die. The user controls the percentage that dies off.	Create a spreadsheet to simulate the "Birthday Problem" (How many people must be in a room for there to be at least a 50% chance that at least two have the same birthday?). Use the same model to answer the question for three people having the same birthday.
Parallelization	Organize resources to simultaneously carry out tasks to reach a common goal.	Based on a set of criteria, break the class into two groups. Have one group read aloud while the other group provides humming background music. The goal is reached, but the whole is better than the individual parts.	Teachers facilitate in planning team project timelines, roles, and assignments and working together to complete components (how do we break up the tasks, what tasks have to be done sequentially and others simultaneously, check ins, meeting deadlines?).	Student teams plan production of a video, including script, props, and roles of the team in producing the video. Identify tasks that will be carried out simultaneously, and milestones where they check in, and plan, and put things together.	Describe the sequence of activities by each of the armies leading to the Battle of Waterloo. Include both physical activities (e.g., recruit troops) and intellectual activities (e.g., pick troop positions).



Bringing Computational Thinking to K-12

continued

TABLE 1: CORE COMPUTATIONAL THINKING CONCEPTS AND CAPABILITIES

CT Concept, Capability	CS	Math	Science	Social Studies	Language Arts
Data collection	Find a data source for a problem area	Find a data source for a problem area, for example, flipping coins or throwing dice	Collect data from an experiment	Study battle statistics or population data	Do linguistic analysis of sentences
Data analysis	Write a program to do basic statistical calculations on a set of data	Count occurrences of flips, dice throws and analyzing results	Analyze data from an experiment	Identify trends in data from statistics	Identify patterns for different sentence types
Data representation	Use data structures such as array, linked list, stack, queue, graph, hash table, etc.	Use histogram, pie chart, bar chart to represent data; use sets, lists, graphs, etc. To contain data	Summarize data from an experiment	Summarize and represent trends	Represent patterns of different sentence types
Problem Decomposition	Define objects and methods; define main and functions	Apply order of operations in an expression	Do a species classification		Write an outline
Abstraction	Use procedures to encapsulate a set of often repeated commands that perform a function; use conditionals, loops, recursion, etc.	Use variables in algebra; identify essential facts in a word problem; study functions in algebra compared to functions in programming; Use iteration to solve word problems	Build a model of a physical entity	Summarize facts; deduce conclusions from facts	Use of simile and metaphor; write a story with branches
Algorithms & procedures	Study classic algorithms; implement an algorithm for a problem area	Do long division, factoring; do carries in addition or subtraction	Do an experimental procedure		Write instructions
Automation		Use tools such as: geometer sketch pad; star logo; python code snippets	Use probeware	Use excel	Use a spell checker
Parallelization	Threading, pipelining, dividing up data or task in such a way to be processed in parallel	Solve linear systems; do matrix multiplication	Simultaneously run experiments with different parameters		
Simulation	Algorithm animation, parameter sweeping	Graph a function in a Cartesian plane and modify values of the variables	Simulate movement of the solar system	Play age of empires; Oregon trail	Do a re-enactment from a story

In attempting to define a classroom culture that would be most conducive to computational thinking, the participants identified strategies or characteristics that could be considered broadly beneficial to any learning experience. These included:

- Increased use by both teachers and students of computational vocabulary where appropriate to describe problems and solutions;
- Acceptance by both teachers and students of failed solution attempts, recognizing that early failure can often put you on the path to a successful outcome;
- Team work by students, with explicit use of:
  - *decomposition* - breaking problems down into smaller parts that may be more easily solved,
  - *abstraction* - simplifying from the concrete to the general as solutions are developed,
  - *negotiation* - groups within the team working together to merge parts of the solution into the whole, and
  - *consensus building* - working to build group solidarity behind one idea or solution.

Samples of 3-level scaffolded CT activities ….Topic: Student Sleep patterns

CT component	Grades PK-2	Grades 3-5	Grades 6-8	Grades 9-12
Problem description Questions should be derived from students. These are examples.	Why is sleep important?	How can we get enough sleep?	What factors impacts sleep?	How does sleep vary across social groups/societies? How can technology be used to help individuals meet their individual sleep needs?
Data collection	Learn about sleep needs of humans and other animals. Match number of hours to animal.	Create and conduct a survey to gather sleep durations of classmates	Sketch (or design and program) an app for gathering data on hours of personal sleep. Locate resources of sleep data.	Employ cloud technologies to crowd gather data from a larger population. Design the survey questions to collect additional data for asking more complex questions of the data.
Data representation	Draw on a clock face the desired amount and the amount sleep children need.	Plot the sleep duration of several classmates for a week on a line or bar chart.	Plot data using different charting formats and select the most effective visual representation strategy. Design the data representation that would allow for easy storage and retrieval of the data. (example: variables for beginning and end of rest periods)	Plan the data storage/variables to enable analysis of data to answer specific questions. Expand the data representation to arrays and more complex data structures.
Data analysis	Make generalizations about the amount of sleep people of various ages need.	Describe patterns of sleep and relate to the day of the week or school schedules.	Describe the statistics you would want to report and share. (Average total sleep per day, standard deviation. Average number of sleep periods per day, including naps). Describe trends, patterns, variations, and/or outliers represented in the chart.	Write formulas to manipulate the data to answer problem sub-questions. Describe trends, patterns, variations, and/or outliers represented in the chart.
Problem decomposition	Break the problem “not enough sleep is unhealthy” into specific ways it might be unhealthy	Develop a plan to personally get enough sleep.	Decompose the problem of tracking different sleep patterns when some students don’t have a single sleep block of time per day. Create a list of sleep suggestions students. Publish along with reasons getting enough sleep is important.	Decompose the main problem question into sub-problems. How does age, work schedule, social group, and other groupings impact sleep? What grouping(s) have the most impact? Write functions/methods with parameters to process the variables within the data.
Abstraction	Summarize the importance of sleep for all creatures.	Draw conclusions about the sleep patterns of their peers.	Describe what is common about all student data and what is variable. Decide what data is important to store and what is not important (depends on your problem definition).	Develop ranking system (to be implemented within program functions) that generalizes the impact of various sleep variables.
Algorithm/procedures	Create a list of steps for getting ready for sleep.	Design a board game that rewards sleep promoting choices and write instructions to play.	Describe the algorithm that would be used to compute the average and variability of data. Write a program for an app that reinforces good sleep habits.	Describe an algorithm to implement the sleep factor rating system.
Automation	Learn about devices that can aid sleep (white noise machines, auto off radio/music devices, alarm clocks, etc.)	Sketch (or design and program) an app to record personal sleep data.	Program an app to collect sleep data from subjects.	Write the program to implement the sleep factor rating system.
Simulation			Given the collected data, describe how a computer might be used to predict what someone’s sleep pattern might be some time in the future.	Create a simulation that enables the user to change various sleep variables (time, time of day, food consumption, noise, etc. whatever elements from the survey that can be changed) to improve their “sleep rating”
Parallelization		Create the app as a team project with distinct roles for teammates.	Student teams plan production of a video, including script, props, and roles of the team in producing the video on sleep.	

ISTE STANDARDS FOR EDUCATORS

Computational Thinking Competencies

Leaders and educators around the world have the enormous responsibility of preparing all students for success in a future where computing power underpins every aspect of the systems we encounter in our daily lives. Ensuring that every student understands and is able to harness the power of computing to improve their success in their personal, academic or professional lives is an ambitious goal. The **ISTE Standards for Educators: Computational Thinking Competencies** is intended to help all educators contribute to making that goal a reality.

In 2019, ISTE will release educator standards specifically for computer science discipline teachers in collaboration with the Computer Science Teachers Association. The Computational Thinking Competencies, however, focus on the educator knowledge, skills and mindsets to integrate computational thinking (CT) across the K-12 content areas and with students of every age. These competencies augment and hone in on the competencies embedded in the ISTE Standards for Students and the ISTE Standards for Educators.

Computational thinking is a powerful ingredient for solving ambiguous, complex and open-ended problems by drawing on principles and practices central to computer science (CS). CT is at the core of computer science and a gateway to sparking student interest and confidence in learning computer science. In these competencies, we use the definition of CS found in the K12 CS Framework, "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their [implementation], and their impact on society," *(Tucker et. al, 2003, p. 6)*, and describe computational thinking as involving designing solutions that leverage the power of computing.

Similarly to how technology is used by educators to deepen content area learning while building digital learning skills, teachers can integrate CT practices in their instruction to introduce computational ideas. This will enhance student content knowledge and build confidence and competence in CT. By integrating computational thinking into the classroom, educators can support students to develop problem-solving and critical-thinking skills, and empower them for success as CS learners and computational thinkers.

ISTE recognizes that the CS concepts framed in current standards and frameworks are not only new to students, but educators as well. Standard 1. Computational Thinking (Learner) is not an expectation of current knowledge, but instead the beginning of a road map to help educators identify strengths and weaknesses, and seek out professional development opportunities to increase their mastery.

This document is not a one-size-fits-all list of expectations, but a recognition that competencies present different opportunities for growth and goal-setting for educators. Educators are doing powerful work to integrate CT across other disciplines to enable students to learn, use and apply CS concepts and CT practices across different contexts. ISTE seeks to help educators recognize where this work is already happening, identify opportunities to make these connections more explicit, and develop new ways to deepen student learning in CS, using computational thinking to drive that work.

1. Computational Thinking (Learner)

Educators continually improve their practice by developing an understanding of computational thinking and its application as a cross-curricular skill. Educators develop a working knowledge of core components of computational thinking: such as decomposition; gathering and analyzing data; abstraction; algorithm design; and how computing impacts people and society. Educators:

a. Set professional learning goals to explore and apply teaching strategies for integrating CT practices into learning activities in ways that enhance student learning of both the academic discipline and CS concepts.

b. Learn to recognize where and how computation can be used to enrich data or content to solve discipline-specific problems and be able to connect these opportunities to foundational CT practices and CS concepts.

c. Leverage CT and CS experts, resources and professional learning networks to continuously improve practice integrating CT across content areas.

d. Develop resilience and perseverance when approaching CS and CT learning experiences, build comfort with ambiguity and open-ended problems, and see failure as an opportunity to learn and innovate.

e. Recognize how computing and society interact to create opportunities, inequities, responsibilities and threats for individuals and organizations.

2. Equity Leader (Leader)

All students and educators have the ability to be computational thinkers and CS learners. Educators proactively counter stereotypes that exclude students from opportunities to excel in computing and foster an inclusive and diverse classroom culture that incorporates and values unique perspectives; builds student self-efficacy and confidence around computing; addresses varying needs and strengths; and addresses bias in interactions, design and development methods. Educators:

a. Nurture a confident, competent and positive identity around computing for every student.
b. Construct and implement culturally relevant learning activities that address a diverse range of ethical, social and cultural perspectives on computing and highlight computing achievements from diverse role models and teams.
c. Choose teaching approaches that help to foster an inclusive computing culture, avoid stereotype threat and equitably engage all students.
d. Assess and manage classroom culture to drive equitable student participation, address exclusionary dynamics and counter implicit bias.
e. Communicate with students, parents and leaders about the impacts of computing in our world and across diverse roles and professional life, and why these skills are essential for all students.

3. Collaborating Around Computing (Collaborator)

Effective collaboration around computing requires educators to incorporate diverse perspectives and unique skills when developing student learning opportunities, and recognize that collaboration skills must be explicitly taught in order to lead to better outcomes than individuals working independently. Educators work together to select tools and design activities and environments that facilitate these collaborations and outcomes. Educators:

a. Model and learn with students how to formulate computational solutions to problems and how to give and receive actionable feedback.
b. Apply effective teaching strategies to support student collaboration around computing, including pair programming, working in varying team roles, equitable workload distribution and project management.
c. Plan collaboratively with other educators to create learning activities that cross disciplines to strengthen student understanding of CT and CS concepts and transfer application of knowledge in new contexts.

4. Creativity & Design (Designer)

Computational thinking skills can empower students to create computational artifacts that allow for personal expression. Educators recognize that design and creativity can encourage a growth mindset and work to create meaningful CS learning experiences and environments that inspire students to build their skills and confidence around computing in ways that reflect their interests and experiences. Educators:

a. Design CT activities where data can be obtained, analyzed and represented to support problem-solving and learning in other content areas.
b. Design authentic learning activities that ask students to leverage a design process to solve problems with awareness of technical and human constraints and defend their design choices.
c. Guide students on the importance of diverse perspectives and human-centered design in developing computational artifacts with broad accessibility and usability.
d. Create CS and CT learning environments that value and encourage varied viewpoints, student agency, creativity, engagement, joy and fun.

5. Integrating Computational Thinking (Facilitator)

Educators facilitate learning by integrating computational thinking practices into the classroom. Since computational thinking is a foundational skill, educators develop every student's ability to recognize opportunities to apply computational thinking in their environment. Educators:

a. Evaluate and use CS and CT curricula, resources and tools that account for learner variability to meet the needs of all students.
b. Empower students to select personally meaningful computational projects.
c. Use a variety of instructional approaches to help students frame problems in ways that can be represented as computational steps or algorithms to be performed by a computer.
d. Establish criteria for evaluating CT practices and content learning that use a variety of formative and alternative assessments to enable students to demonstrate their understanding of age-appropriate CS and CT vocabulary, practices and concepts.

2016

ISTE STANDARDS FOR STUDENTS

1. Empowered Learner

Students leverage technology to take an active role in choosing, achieving and demonstrating competency in their learning goals, informed by the learning sciences. Students:

a. articulate and set personal learning goals, develop strategies leveraging technology to achieve them and reflect on the learning process itself to improve learning outcomes.
b. build networks and customize their learning environments in ways that support the learning process.
c. use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.
d. understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

2. Digital Citizen

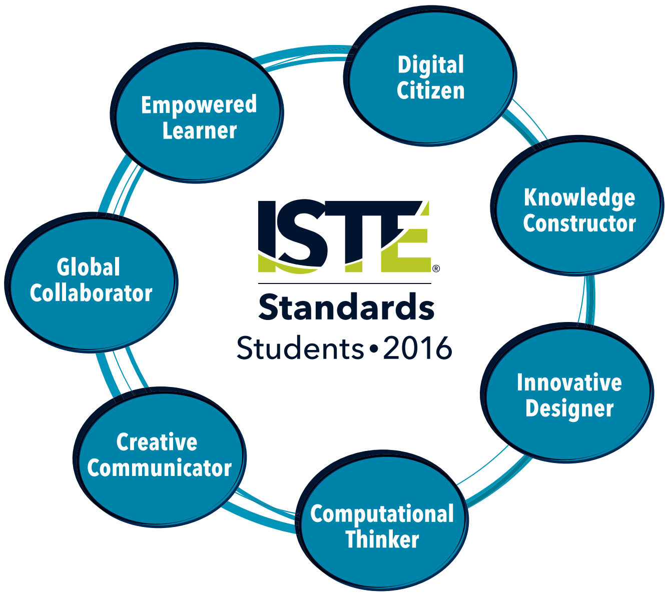
Students recognize the rights, responsibilities and opportunities of living, learning and working in an interconnected digital world, and they act and model in ways that are safe, legal and ethical. Students:

a. cultivate and manage their digital identity and reputation and are aware of the permanence of their actions in the digital world.
b. engage in positive, safe, legal and ethical behavior when using technology, including social interactions online or when using networked devices.
c. demonstrate an understanding of and respect for the rights and obligations of using and sharing intellectual property.
d. manage their personal data to maintain digital privacy and security and are aware of data-collection technology used to track their navigation online.

3. Knowledge Constructor

Students critically curate a variety of resources using digital tools to construct knowledge, produce creative artifacts and make meaningful learning experiences for themselves and others. Students:

a. plan and employ effective research strategies to locate information and other resources for their intellectual or creative pursuits.
b. evaluate the accuracy, perspective, credibility and relevance of information, media, data or other resources.
c. curate information from digital resources using a variety of tools and methods to create collections of artifacts that demonstrate meaningful connections or conclusions.
d. build knowledge by actively exploring real-world issues and problems, developing ideas and theories and pursuing answers and solutions.



4. Innovative Designer

Students use a variety of technologies within a design process to identify and solve problems by creating new, useful or imaginative solutions. Students:

a. know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.
b. select and use digital tools to plan and manage a design process that considers design constraints and calculated risks.
c. develop, test and refine prototypes as part of a cyclical design process.
d. exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.

5. Computational Thinker

Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions. Students:

a. formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.
b. collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.
c. break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.
d. understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

6. Creative Communicator

Students communicate clearly and express themselves creatively for a variety of purposes using the platforms, tools, styles, formats and digital media appropriate to their goals. Students:

a. choose the appropriate platforms and tools for meeting the desired objectives of their creation or communication.
b. create original works or responsibly repurpose or remix digital resources into new creations.
c. communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models or simulations.
d. publish or present content that customizes the message and medium for their intended audiences.

7. Global Collaborator

Students use digital tools to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally. Students:

a. use digital tools to connect with learners from a variety of backgrounds and cultures, engaging with them in ways that broaden mutual understanding and learning.
b. use collaborative technologies to work with others, including peers, experts or community members, to examine issues and problems from multiple viewpoints.
c. contribute constructively to project teams, assuming various roles and responsibilities to work effectively toward a common goal.
d. explore local and global issues and use collaborative technologies to work with others to investigate solutions.