Play Code Learn

DINOSAUR COMMANDS

Formative Assessment

All handouts are A4 for printing.

Copyright ByteEd

Links to Massachusetts K-12 Computer Science Curriculum

Progressive Learning with Play Code Learn: Dinosaur Series

Dinosaur Commands has been designed to support learning of digital technologies in the classroom. It is the third kit in a series which link to the progression of the curriculum.

Elementary School						Middle School			High school			
K	1	2	3	4	5	6	7	8	9	10	11	12
Primary						Intermediate/Lower Secondary			Senior Secondary			
1	2	3	4	5	6	7	8	9	10	11	12	13
Dinosaur Steps												
	Dinosaur Loops											
		Dinosaur Commands										

Learning Progression				
Grade Spans	**Strands**			
K-2 3-5 6-8 9-12	**Computing and Society [CAS]** a. Safety and Security b. Ethics and Laws c. Interpersonal and Societal Impact	**Digital Tools and Collaboration [DTC]** a. Digital Tools b. Collaboration and Communication c. Research	**Computing Systems [CS]** a. Computing Devices b. Human and Computer Partnerships c. Networks d. Services	**Computational Thinking [CT]** a. Abstraction b. Algorithms c. Data d. Programming and Development e. Modeling and Simulation
Practices				
Connecting, Creating, Abstracting, Analyzing, Communicating, Collaborating, Research				

https://www.doe.mass.edu/stem/dlcs/?section=512

Links to curriculum

Computer Science: Dinosaur Commands links to all strands in the Computer Science curriculum. (see opposite)

Science & Technology: Through the STEM approach to learning, the use of Dinosaur Commands can also integrate with other subject curricula (dependant on the teaching in the classroom).

Digital Literacy and Computer Science Practices 3-5

Creating, Connecting, Abstracting, Analyzing, Communicating, Collaborating, Researching.

With increased maturity, students in third through fifth grade are able to engage in learning in ways that are both more systematic and creative. Upper elementary is a critical time to engage students in the DLCS practices. Students' **capabilities as creators and problem solvers build on their experiences in K-2.** They **continue to develop concepts through exploration, discovery, and creativity** with the guidance, support, and encouragement of their educator. Standards for this grade span allow teacher flexibility in deciding when students are ready to use technology.

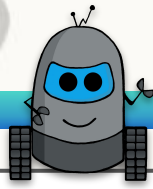
Digital Literacy and Computer Science Practices 6-8

Creating, Connecting, Abstracting, Analyzing, Communicating, Collaborating, Researching.

By the time students reach middle school, they should have had numerous experiences in using technology to **create artifacts and solve problems.** Active engagement of middle school students with the practices is critical: students generally make up their minds about whether they identify with science and engineering by the time they leave grade 8. Students should have opportunities to **develop the skills necessary for a meaningful progression of development in order to engage in reasoning,** which is critical to success in civic life, post-secondary education, and career.

Highlighted in **bold** are the practices that Dinosaur Commands supports.

Copyright ByteEd



Links to Massachusetts **K-2 Computer Science** Curriculum

Computing & society (CAS)

Safety and Security [K-2.CAS.a]

1. Demonstrate proper ergonomics (e.g., body position, stretching) when using devices.
2. Use electrical devices safely and in moderation (e.g., unplug devices by pulling the plug rather than the cord, do not mix water/food and electric devices, avoid gaming and walking).
3. Care for devices appropriately (e.g., handling devices gently, completely shutting down devices when not in use, storing devices in the appropriate container).
4. Explain that a password helps protect the privacy of information.

Digital Tools & collaboration (DTC)

Digital Tools [K-2.DTC.a]

1. Operate a variety of digital tools (e.g., open/close, find, save/print, navigate, use input/output devices).

Computing Systems (CS)

Computing Devices [K-2CS.a]

1. Identify different kinds of computing devices in the classroom and other places (e.g., laptops, tablets, smart phones, desktops).
2. Identify visible components of computing devices (e.g., keyboard, screen, monitor, printer, pointing device).
3. Explain that computing devices function when applications, programs, or commands are executed.
4. Operate a variety of computing systems (e.g., turn on, use input/output devices such as a mouse, keyboard, or touch screen; find, navigate, launch a program).

Human and Computer Partnerships [K-2.CS.b]

1. Explain that computing devices are machines that are not alive but can be used to help humans with tasks.
2. Recognize that some tasks are best completed by humans and others by computing devices (e.g., a human might be able to rescue someone in a normal environment, but robots would be better to use in a dangerous environment).
3. Recognize that different tools can solve the same problem (e.g., pen and paper, calculators, and smartphones can all be used to solve simple mathematical problems).

Computational Thinking (CT)

Abstraction [K-2.CT.a]

1. List the attributes of a common object, for example, cars have a color, type (e.g., pickup, van, sedan), number of seats, etc.

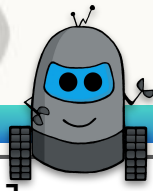
Algorithms [K-2.CT.b]

1. Define an algorithm as a sequence of defined steps.
2. Create a simple algorithm, individually and collaboratively, without using computers to complete a task (e.g., making a sandwich, getting ready for school, checking a book out of the library).
3. Enact an algorithm using tangible materials (e.g., manipulatives, your body) or present the algorithm in a visual medium (e.g., storyboard).

Programming and Development [K-2.CT.d]

1. Define a computer program as a set of commands created by people to do something.
2. Explain that computers only follow the program's instructions.
3. Individually or collaboratively, create a simple program using visual instructions or tools that do not require a textual programming language (e.g., "unplugged" programming activities, a block-based programming language).

Copyright ByteEd



Links to Massachusetts **3-5 Computer Science** Curriculum

Computing & society (CAS)

Safety and Security [3-5.CAS.a]

1. Describe how to use proper ergonomics (e.g., body position, lighting, positioning of equipment, taking breaks) when using devices.
3. Explain the proper use and operation of security technologies (e.g., passwords, virus protection software, spam filters, popup blockers, cookies).
4. Explain the proper use and operation of security technologies (e.g., passwords, virus protection software, spam filters, popup blockers, cookies).

Computing Systems (CS)

Computing Devices [3-5.CS.a]

1. Identify a broad range of computing devices (e.g., computers, smart phones, tablets, robots, e-textiles) and appropriate uses for them.
2. Describe the function and purpose of various input and output devices (e.g., monitor, keyboard, speakers, controller, probes, sensors, Bluetooth transmitters, synthesizers).
3. Demonstrate an appropriate level of proficiency (connect and record data, print, send command, connect to Internet, search) in using a range of computing devices (e.g., probes, sensors, printers, robots, computers).
4. Identify and solve simple hardware and software problems that may occur during everyday use (e.g., power, connections, application window or toolbar).
5. Describe the differences between hardware and software.

Human and Computer partnerships [3-5.CS.b]

1. Compare and contrast human and computer performance on similar tasks (e.g., sorting alphabetically, finding a path across a cluttered room) to understand which is best suited to the task
3. Explain advantages and limitations of technology (e.g., a spell-checker can check thousands of words faster than a human could look them up, however, a spell-checker might not know whether 'underserved' is correct or if the author's intent was to type 'undeserved').

Computational Thinking (CT)

Abstraction [3-5.CT.a]

1. Use numbers or letters to represent information in another form (e.g., secret codes, Roman numerals, abbreviations).
1. Organise information in different ways to make it more useful/relevant (e.g. sorting, tables)
2. Make a list of sub-problems to consider, while addressing a larger problem.

Algorithms [3-5.CT.b]

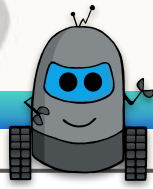
1. Define an algorithm as a sequence of instructions that can be processed by a computer.
2. Recognize that different solutions exist for the same problem (or sub-problem).
3. Use logical reasoning to predict outcomes of an algorithm.
4. Individually and collaboratively create an algorithm to solve a problem (e.g., move a character/robot/person through a maze).
5. Detect and correct logical errors in various algorithms (e.g., written, mapped, live action, or digital).

Programming and Development [3-5.CT.d]

1. Individually and collaboratively create, test, and modify a program in a graphical environment (e.g., block-based visual programming language).
2. Use arithmetic operators, conditionals and repetition in programs.
3. Use interactive debugging to detect and correct simple program errors.
4. Recognize that programs need known starting values (e.g., set initial score to zero in a game).

Formative Assessment printables will link to the **Computational Thinking Standards.**

Copyright ByteEd



Links to Massachusetts **6-8 Computer Science** Curriculum

Computing & society (CAS)

Interpersonal and societal impact[6-8.CAS.b]

1. Describe current event and emerging technologies in computing and the effects they may have on education, the workplace, individuals, communities & global societies.

Digital Tools & Collaboration (DTC)

Digital Tools [6-8.DTC.a]

1. Identify and explain the strengths, weaknesses and capabilities of a variety of digital tools.

Computing Systems (CS)

Human and Computer Partnerships [6-8.CS.b]

1. Explain why some problems can be solved more easily by computers or humans based on a general understanding of types of tasks at which each excels.
2. Describe how humans and machines interact to solve problems that cannot be solved by either alone (e.g., "big data" experiments that involve drawing conclusions by analyzing vast amounts of data).

Computational Thinking (CT)

Abstraction [6-8.CT.a]

1. Describe how data is abstracted by listing attributes of everyday items to represent, order and compare those items (e.g., street address as an abstraction for locations; car make, model, and license plate number as an abstraction for cars).
2. Define a simple function that represents a more complex task/problem and can be reused to solve similar tasks/problems.
3. Use decomposition to define and apply a hierarchical classification scheme to a complex system, such as the human body, animal classification, or in computing.

Algorithms [6-8.CT.b]

1. Design solutions that use repetition and conditionals.
2. Use logical reasoning to predict outputs given varying inputs.
3. Individually and collaboratively, decompose a problem and create a sub-solution for each of its parts (e.g., video game, robot obstacle course, making dinner).
4. Recognize that more than one algorithm can solve a given problem.
5. Recognize that boundaries need to be taken into account for an algorithm to produce correct results.

Data [6-8.CT.c]

1. Demonstrate that numbers can be represented in different base systems (e.g., binary, octal, and hexadecimal) and text can be represented in different ways [e.g., American Standard Code for Information Interchange (ASCII)].

Programming and Development [3-5.CT.d]

1. Individually and collaboratively compare algorithms to solve a problem, based on a given criteria (e.g., time, resource, accessibility).
2. Use functions to hide the detail in a program.
3. Create a program, individually and collaboratively, that implements an algorithm to achieve a given goal.
4. Implement problem solutions using a programming language, including all of the following: looping behavior, conditional statements, expressions, variables, and functions.
5. Trace programs step-by-step in order to predict their behavior.
6. Use an iterative approach in development and debugging to understand the dimensions of a problem clearly.

Copyright Bytes

STEM & Technology learning links to Massachusetts 2016 Science & Technology Learning Framework Guiding Principles:
(https://www.doe.mass.edu/frameworks/scitech/2016-04.pdf)

Relevance

- Clear links to the society of today through the use of programming and Augmented Reality and through the exploration of how computer scientists & programmers use CT concepts in their everyday lives.
- Transference of skill and knowledge from other curriculum areas e.g. Maths.
- Enabling curiosity, rectifying misconceptions, inspiring wonder, making meaning.

Rigour

- Increased understanding and development of technological literacy.
- Develops problem solving skills and reflection.
- Expands the ability to communicate and collaborate with others.
- Explores critical and logical thinking.

Coherence

- Links to the core subject areas of mathematics and literacy.
- Clear learning progression in the kits aiding the assessment of learning.

Standards (**some examples** of links with Play Code Learn teaching & learning):

Earth & Space Sciences

- 4-ESS1-1. Use evidence from a given landscape that includes simple landforms and rock layers to support a claim about the role of erosion or deposition in the formation of the landscape over long periods of time.
-

Earth's Systems

- 2-ESS2-2. Map the shapes and types of landforms and bodies of water in an area.
- 3-ESS2-2. Obtain and summarize information about the climate of different regions of the world to illustrate that typical weather conditions over a year vary by region.
- 4-ESS2-2. Analyze and interpret maps of Earth's mountain ranges, deep ocean trenches, volcanoes, and earthquake epicenters to describe patterns of these features and their locations relative to boundaries between continents and oceans.

Life Science

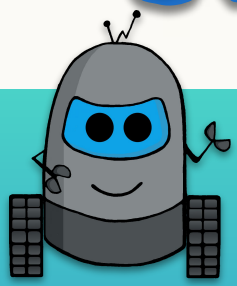
- 2-LS4-1. Use texts, media, or local environments to observe and compare (a) different kinds of living things in an area, and (b) differences in the kinds of living things living in different types of areas.
- 3-LS1-1. Use simple graphical representations to show that different types of organisms have unique and diverse life cycles. Describe that all organisms have birth, growth, reproduction, and death in common but there are a variety of ways in which these happen.
- 3-LS3-2. Distinguish between inherited characteristics and those characteristics that result from a direct interaction with the environment. Give examples of characteristics of living organisms that are influenced by both inheritance and the environment.
- 3-LS4-1. Use fossils to describe types of organisms and their environments that existed long ago and compare those to living organisms and their environments. Recognize that most kinds of plants and animals that once lived on Earth are no longer found anywhere.
- 3-LS4-4. Analyze and interpret given data about changes in a habitat and describe how the changes may affect the ability of organisms that live in that habitat to survive and reproduce

Technology & Engineering

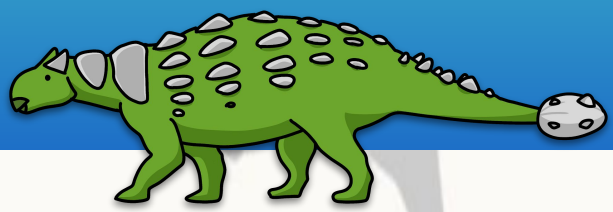
- 1.K-2-ETS1-2. Generate multiple solutions to a design problem and make a drawing (plan) to represent one or more of the solutions.
- 3.3-5-ETS1-2. Generate several possible solutions to a given design problem. Compare each solution based on how well each is likely to meet the criteria and constraints of the design problem.

Copyright ByteEd

Play Code Learn



DINOSAUR COMMANDS



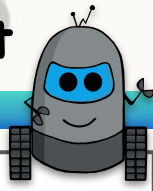
Assessment:

Teacher Assessment

Copyright ByteEd

DINOSAUR COMMANDS

Teacher Assessment



Student Name:	Date:

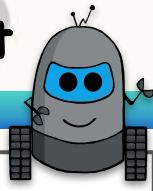
Computational Thinking: K-2

Creating, **Connecting**, **Abstracting**, **Analysing**, **Communicating**, **Collaborating**, Researching.

Student ...	✓	Standard Criteria
		Abstraction
		Can list the attributes for a object, *e.g. dinosaurs have a colour, type, number of legs, etc.*
		Algorithms
		Can explain that an algorithm is a sequence of steps.*(step by step instructions)*
		Can create a simple algorithm to complete a task that does not use computers.
		Can follow an algorithm using tangible materials *e.g. dinosaur steps command cards.*
		Programming & Development
		Can explain that a program is a set of commands, created by humans, for a digital device to complete a task.
		Can explain that a computer can only follow the program's instructions.
		Can create a simple program using visual instructions *e.g. ByteEd App.*

Copyright ByteEd

DINOSAUR COMMANDS Teacher Assessment



Student Name:	Date:

Computational Thinking: 3-5

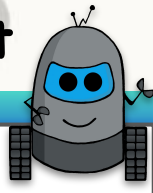
Creating, **Connecting**, **Abstracting**, **Analysing**, **Communicating**, **Collaborating**, Researching.

Student ...	✓	Standard Criteria
		Abstraction
		Can use numbers or letters to represent information in a another form *e.g secret codes*.
		Algorithms
		Can explain that an algorithm is a sequence of instructions that can be processed by a computer.
		Recognises that different solutions can exist for the same problem.
		Uses logical reasoning to predict the outcomes of an algorithm.
		Can create an algorithm to solve a problem *e.g. Chompy's Challenges*.
		Can detect & correct errors in various algorithms *(debugging)*.
		Programming & Development
		Can create, test & modify a program in a graphical environment, *e.g. ByteEd App*.
		Can use debugging to detect and correct simple program errors.

Copyright ByteEd

DINOSAUR COMMANDS

Teacher Assessment



Student Name: Date:

Computational Thinking: 6-8

Creating, Connecting, Abstracting, Analysing, Communicating, Collaborating, Researching.

Student ...

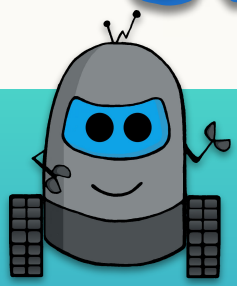
✓	Standard Criteria
	Abstraction
	Can abstract data by listing and ordering attributes of everyday items and comparing them.
	Can define and reuse simple functions that represent a more complex task/problem.
	Use decomposition to define & apply a hierarchical classification scheme *e.g.dinosaur classification*
	Algorithms
	Can design solutions that use repetition & conditionals.
	Use logical reasoning to predict outputs given varying inputs.
	Can decompose a problem and create a sub-solution for each of its parts *e.g.break down an algorithm challenge*
	Can recognise that more than one algorithm can solve a given problem.
	Can recognise that boundaries need to be taken into account for an algorithm to produce correct results.
	Data
	Can demonstrate that numbers can be represented in different base systems *e.g. binary numbers*
	Programming & Development
	Can compare algorithms to solve a problem, based on a given criteria, *e.g. time, efficiency*
	Can use functions to hide the detail in a program.
	Can create a program that implements an algorithm to achieve a given goal.
	Can implement solutions using a programming language that uses; looping, conditionals, expressions, variables & functions.
	Can trace programs step by step in order to predict their behaviour.
	Can use an iterative approach in developing and debugging a program.

DINOSAUR COMMANDS Teacher Assessment

Student Name											

Copyright ByteEd

Play Code Learn



DINOSAUR COMMANDS

Assessment:

Student Assessment

Also available are **lesson exit statements/reflections** for each lesson in the teaching module for Dinosaur Commands.

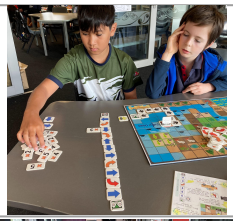
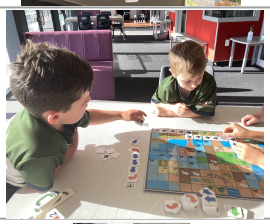
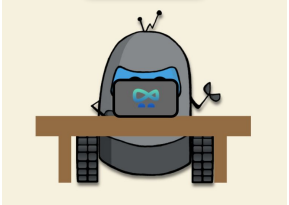
Copyright ByteEd

DINOSAUR COMMANDS

Student Reflection

Student Name: Date:

Computational Thinking: K-2

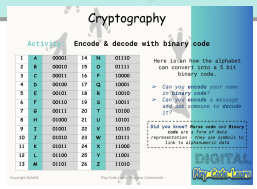
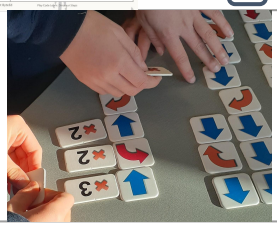
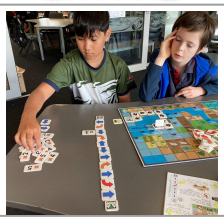
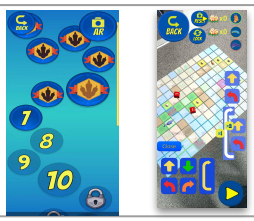
				I can...
				...list attributes.
				...understand an algorithm is a set of step by step instructions.
				...create a simple algorithm (step by step).
				...follow an algorithm.
				...understand a program is a set of commands for a computer to follow.
				...explain a computer can only follow a program's instructions.
				...create a simple program.

Copyright ByteEd

DINOSAUR COMMANDS Student Reflection

Student Name: Date:

Computational Thinking: 3-5

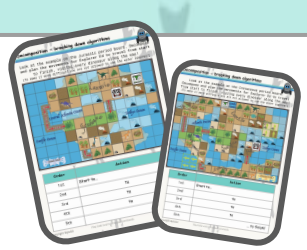
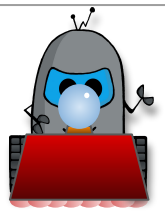
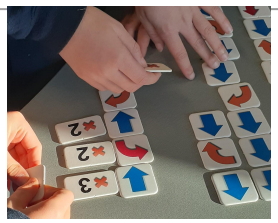
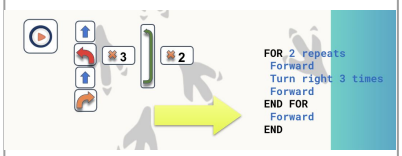
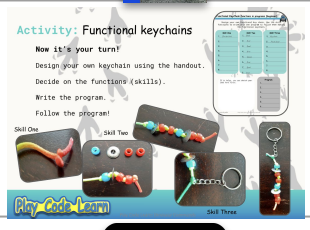
				I can...
				...use numbers or letters to make secret codes.
				...explain that an algorithm is a list of instructions.
				...understand that there can be more than one solution to a problem.
				...predict what will happen with an algorithm.
				...can create algorithms to solve problems.
				...can detect & correct errors (bugs) in algorithms.
				...can create, test & change programs.
				...debug errors in programs.

Copyright ByteEd

DINOSAUR COMMANDS — Student Reflection

Student Name: Date:

Computational Thinking: 6-8 *Algorithms & Programming*

				I can...
				...decompose a problem & create a solution for each part.
				...use logical thinking to predict outputs in algorithms & programs.
				...understand that there can be more than one solution to a problem.
				...compare algorithms to solve a problem.
				...can create a program based on an algorithm.
				...can create algorithms & programs that use repetition *(looping)*.
				...can use functions in a program.
				...use an iterative approach when developing & debugging algorithms and/or programs.