Unit 6: Physical Computing

Content Area: **Science**
Course(s):
Time Period: **MP4**
Length:
Status: **Published**

NJSLS - Science

8.1.12.CS.2	Model interactions between application software, system software, and hardware.
8.1.12.CS.3	Compare the functions of application software, system software, and hardware.
8.1.12.DA.1	Create interactive data visualizations using software tools to help others better understand real world phenomena, including climate change.
8.1.12.DA.5	Create data visualizations from large data sets to summarize, communicate, and support different interpretations of real-world phenomena.
8.1.12.DA.6	Create and refine computational models to better represent the relationships among different elements of data collected from a phenomenon or process.
8.1.12.AP.5	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

Rationale and Transfer Goals

Students will be able to demonstrate awareness of simple programming concepts such as sequencing, repton, variables, and selection by writing short programs. So far students will have used block-based programming and JavaScript in the code.org App Studio. Students will learn how to translate their knowledge of programming from those languages to that of Python. Using Python, students will be able to use simple control flow statements and variables to use a variety of logical, arithmetic, and comparison operators. These will then use the GPiO pins on the Raspberry Pi to make use of LEDs and buttons. Finally, they will end the year by programming a Minecra game and building their controllers to play.

Enduring Understandings

- Creave development can be an essential process for creating a computational artifact.
- People use computer programs to process information to gain insight and knowledge.
- Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.
- Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems.
- People write programs to execute algorithms.
- Programming uses mathematical and logical concepts.

Essential Questions

- How does software interact with hardware?
- How can computers communicate information with simple hardware outputs?
- How can programs be made to repeat tasks?
- How can computers sense and respond to their environment?
- How can complex real-world information be represented in code?
- How can simple hardware be used to develop innovative new products?

Content - What will students know?

- Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
- Use flowcharts and/or pseudocode to address complex problems as algorithms.
- Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
- Incorporate existing code, media, and libraries into original programs, and give a ribbon.
- Systemacally test and refine programs using a range of test cases.
- Create named variables that represent different data types and perform operations on their values.
- Document programs to make them easier to follow, test, and debug.
- Design projects that combine hardware and software components to collect and exchange data.
- Systematically identify and fix problems with computing devices and their components.
- Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.
- Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- Seek and incorporate feedback from team members and users to refine a solution that meets user needs.

Skills - What will students be able to do?

- Idenfy computing innovations within a given field
- For a given device, articulate the likely inputs and outputs
- Suggest improvements to help a device beer solve a specific problem
- Compare and contrast multiple ways to take input
- Describe the elements of an event handler
- Model different methods of taking user input
- Create a user interface composed of multiple design elements
- Use event handlers to respond to user input
- Explain the difference between the "click", "change", and "input" events, and identify scenarios for each
- Use a geer to get the current content of a UI element
- Use a seer to change the content or properties of a UI element
- Write programs that change multiple elements on a single screen instead of changing screens
- Connect and troubleshoot external devices
- Turn on and off an LED with code

- Access a specific location in a list using its index
- Arculate the difference between the length of a list and the index of its last value
- Access items in an array by index
- Apply the RGB color model to LEDs
- Understand how to use a for loop as a way to repeat a set of code a certain number of times
- Trace the execution of a for-loop
- Given a for loop, predict how many times it will repeat
- Use for loops to process through the color Leds and do something to all the color LEDs
- Use for loops to process a list
- Use a med loop to write a non-blocking infinite loop
- Replicated a for loop with a med loop
- Use event handlers to take user input
- Output simple information on a physical device
- Use a loop to repeat instructions
- Prototype a program that integrates software and hardware
- Aach an event handler to a hardware input
- ● Choose the appropriate event for a given scenario
- Aach an event handler to a hardware input
- Choose the appropriate event for a given scenario
- Develop apps that take input through analog sensors
- Independently scope the features of a piece of software
- Prototype a physical computing device
- Implement a plan for developing a piece of software that integrates hardware inputs and outputs

Activities - How will we teach the content and skills?

- App Lab
- Maker Toolkit
- Project
- Python Programming
- Raspberry Pi
- Minecraft
- Breadboards with LEDs and Buttons
- Unplugged
- Research

Evidence/Assessments - How will we know what students have learned?

- Rubrics for projects from code.org
- Students will be able to control an LED turning on and off. Students will be able to use a button to perform an event in Minecraft.

- Students will be able to create a controller using user-made buttons to make a character turn le, right, forward, and backward in Minecraft.
- Students will know the parts of a Raspberry Pi and be able to connect a breadboard to it

Spiraling for Mastery

Content or Skill for this Unit	Spiral Focus from Previous Unit	Instructional Activity
• People evaluate and select algorithms based on performance, reusability, and ease of implementation. Knowledge of common algorithms improves how people develop software, secure data, and store information. • Data structures are used to manage program complexity. Programmers choose data structures based on functionality, storage, and performance tradeoffs. • Data structures are used to manage program complexity. Programmers choose data structures based on functionality, storage, and performance tradeoffs. • Programmers consider tradeoffs related to implementation, readability, and program performance when selecting and combining control structures. • Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program; combinations of data and	• Algorithms affect how people interact with computers and the way computers respond. People design algorithms that are generalizable to many situations. Readable algorithms are easier to follow, test, and debug. • Programmers create variables to store data values of selected types. A meaningful identity is assigned to each variable to access and perform operations on the value by name. Variables enable the flexibility to represent different situations, process different sets of data, and produce varying outputs. • Programmers select and combine control structures, such as loops, event handlers, and conditionals, to create more complex program behavior. • Programs use procedures to organize code, hide implementation details, and make code easier to reuse. Procedures can be repurposed in new programs. Defining parameters for procedures can generalize behavior and increase reusability. • People design meaningful soluons for others by defining a problem’s	• Learning to Program with Python • Physical Computing with Python • Input and Output Controls • Minecraft • Build a Controller for Minecraft

procedures; or independent, but interrelated, programs. Modules allow for beer management of complex tasks. • Diverse teams can develop programs with a broad impact through careful review and by drawing on the strengths of members in different roles. Design decisions often involve tradeoffs. The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program. Systematic analysis is critical for identifying the effects of lingering bugs	criteria and constraints, carefully considering the diverse needs and wants of the community, and testing whether criteria and constraints were met.	

Key Resources

- Raspberry Pi Curriculum
- Raspberry Pi Resources
- Minecraft with Raspberry Pi and Python
- Learning Python
- Skulpt Python in a Browser
- Unit 6- Code.org Computer Discoveries Curriculum

Career Readiness, Life Literacies, & Key Skills

9.4.12.CI	Creativity and Innovation Collaboration with individuals with diverse experiences can aid in the problem-solving process, particularly for global issues where diverse solutions are needed.
9.4.12.CT.2	Explain the potential benefits of collaborating to enhance critical thinking and problem solving (e.g., 1.3E.12profCR3.a).
CAEP.9.2.12.C.7	Examine the professional, legal, and ethical responsibilities for both employers and employees in the global workplace.