Unit 5: Lists, Loops and Traversals

Content Area: **Mathematics**
Course(s): **Generic Course**
Time Period: **Semester 1**
Length: **3 weeks**
Status: **Published**

Standards

CS.9-12.8.1.12.AP.1	Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.2	Create generalized computational solutions using collections instead of repeatedly using simple variables.
CS.9-12.8.1.12.AP.3	Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
CS.9-12.8.1.12.AP.4	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
CS.9-12.8.1.12.AP.5	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
CS.9-12.8.1.12.AP.7	Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
CS.9-12.8.1.12.AP.8	Evaluate and refine computational artifacts to make them more usable and accessible.
CS.9-12.8.1.12.AP.9	Collaboratively document and present design decisions in the development of complex programs.
	Individuals evaluate and select algorithms based on performance, reusability, and ease of implementation.
	Programmers choose data structures to manage program complexity based on functionality, storage, and performance trade-offs.
	Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. Modules allow for better management of complex tasks.
	Complex programs are developed, tested, and analyzed by teams drawing on the members' diverse strengths using a variety of resources, libraries, and tools.
	Trade-offs related to implementation, readability, and program performance are considered when selecting and combining control structures.

Essential Questions

How is abstraction part of coding?

How do we use strings in our programs?

How is iteration used to make complex programs more simple?

How can we manage memory with lists?

How do simulations help us understand problems and plan for the future?

How do people develop, test, and debug programs?

Enduring Understanding

- 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.
- To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.
- 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.
- 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.

Knowledge and Skills

Students learn to build apps that use and process lists of information. Like the previous unit, students learn the core concepts of lists, loops, and traversals through a series of EIPM lesson sequences. Later in the unit, students are introduced to tools that allow them to import tables of real-world data to help further power the types of apps they can make. At the conclusion of the unit, students complete a week-long project in which they must design an app around a goal of their choosing that uses one of these data sets.

Transfer Goals

Creating content is a cyclical process.

Collaborating on projects is not usually easier, but produces better results.

Resources

1. Various YouTube videos that visually explain concepts and ideas.

2. Various widgets found on code.org.

3. Test banks created on Edulastic and code.org

4. Use of Google Classroom, Google Slides, Google Docs and Google Sheets