Unit 7: Parameters, Return, and Libraries

Content Area: **Mathematics**
Course(s): **Generic Course, AP Comp Sci A**
Time Period: **Semester 2**
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.
CS.9-12.AP	Algorithms & Programming
	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 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 can calling procedures make our programs more effecient?
- Why do we develop procedures?
- How can we use the libraries of any programming language we learn to our benefit?

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.
- 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 how to design clean and reusable code that can be shared with a single classmate or the entire world. In the beginning of the unit, students are introduced to the concepts of parameters and return, which allow for students to design functions that implement an algorithm. In the second half of the unit, students learn how to design libraries of functions that can be packaged up and shared with others. The unit concludes with students designing their own small library of functions that can be used by a classmate.

Transfer Goals

Procedures exist in all high-level programming languages.

Procedures are an important part of abstraction.

Procedures break down big ideas into small parts.

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