Unit III: Algorithms & Programming

Content Area: **Business**
Course(s): **Introduction to Computer Science and Programing**
Time Period: **4 weeks**
Length: **Weeks**
Status: **Published**

Unit Overview

In this unit, students will learn the fundamentals of programming languages, and apply these skills to create programs that perform useful tasks while applying general principles of program design.

Transfer

Students will be able to independently use their learning to...

- explain the need for unambiguous and precise languages for computer programs
- use a programming language to accomplish simple procedural tasks
- define and use their own procedures (or functions) in a programming language
- use simple 'for' loops to add repetition to their code
- create and assign values to variables
- use counter variables inside a 'for' loop
- distinguish between global and local variables, and use variables to track the score in a simple game
- write and use if statements
- create a useful graphical program, such as a painting program or scrolling image viewer
- create simple animations or games
- explore further concepts such as while loops, arrays, and functions with parameters

Meaning

Understandings

Students will understand that...

- Variables, if statements and loops are fundamental to all programming languages.
- General principles of structuring a program through procedural abstraction (defining functions that perform a useful task) are applicable to all programming languages.
- Principles of programming can be applied to create useful programs and apps.

Essential Questions

Students will keep considering...

- Why do we need algorithms?
- How is designing an algorithm to solve a problem different from other kinds of problem solving?
- How do you design a solution for a problem so that is programmable?
- What does it mean to be a "creative" programmer?
- How do programmers collaborate?
- What are the core features of most programming languages?
- How does programming enable creativity and expression?
- Which practices and strategies will help me as I write programs?
- How do software developers manage complexity and scale?
- How can programs be organized so that common problems only need to be solved once?
- How can I build on previous solutions to create even more complex behavior?

Application of Knowledge and Skill

Students will know...

Students will know...

- Variables, if statements and loops are fundamental to all programming languages.
- General principles of structuring a program through procedural abstraction (defining functions that perform a useful task) are applicable to all programming languages.
- Principles of programming can be applied to create useful programs and apps.

Students will be skilled at...

Students will be skilled at...

- using variables, if statements and loops while creating a program using a programming language.
- applying general principles of structuring a program through procedural abstraction to avoid repeating code.
- utilizing the principles of programming to create useful programs and apps.

Academic Vocabulary

- Imperative/procedural programming
- Variables
- Global variable
- Local variable
- For loops
- If statements
- While loops
- Arrays
- Functions/procedures
- Bug, debugging
- Comment
- Indentation
- Whitespace
- Parameter
- Expression
- Boolean

Learning Goal 1

Students will understand and apply the fundamentals of programming languages

CRP.K-12.CRP2	Apply appropriate academic and technical skills.
TECH.8.1.12.A	Technology Operations and Concepts: Students demonstrate a sound understanding of technology concepts, systems and operations.
TECH.8.1.12.A.CS1	Understand and use technology systems.
TECH.8.1.12.B.2	Apply previous content knowledge by creating and piloting a digital learning game or tutorial.
TECH.8.1.12.B.CS2	Create original works as a means of personal or group expression.
TECH.8.2.12.C.4	Explain and identify interdependent systems and their functions.
TECH.8.2.12.E	Computational Thinking: Programming: Computational thinking builds and enhances problem solving, allowing students to move beyond using knowledge to creating knowledge.
TECH.8.2.12.E.3	Use a programming language to solve problems or accomplish a task (e.g., robotic functions, website designs, applications, and games).
TECH.8.2.12.E.4	Use appropriate terms in conversation (e.g., troubleshooting, peripherals, diagnostic software, GUI, abstraction, variables, data types and conditional statements).
TECH.8.2.12.E.CS1	Computational thinking and computer programming as tools used in design and engineering.

Target 1

Students will understand the need for unambiguous and precise languages for computer programs

Target 2

Students will use a programming language to accomplish simple procedural tasks (e.g. for turtle graphics)

Target 3

Students will define and use their own procedures (or functions) to create and give a name to a group of commands for easy and repeated use in their code

Target 4

Students will understand that complex programs are constructed from simple concepts

Target 5

Students will learn to use simple 'for' loops to add repetition to their code

Learning Goal 2

Students will learn general principles of program design applicable to any programming language

CRP.K-12.CRP2	Apply appropriate academic and technical skills.
TECH.8.1.12	Educational Technology: All students will use digital tools to access, manage, evaluate, and synthesize information in order to solve problems individually and collaborate and to create and communicate knowledge.
TECH.8.1.12.A	Technology Operations and Concepts: Students demonstrate a sound understanding of technology concepts, systems and operations.
TECH.8.1.12.F.CS2	Plan and manage activities to develop a solution or complete a project.
TECH.8.2.12.C	Design: The design process is a systematic approach to solving problems.
TECH.8.2.12.C.4	Explain and identify interdependent systems and their functions.
TECH.8.2.12.E.1	Demonstrate an understanding of the problem-solving capacity of computers in our world.

TECH.8.2.12.E.3	Use a programming language to solve problems or accomplish a task (e.g., robotic functions, website designs, applications, and games).
TECH.8.2.12.E.4	Use appropriate terms in conversation (e.g., troubleshooting, peripherals, diagnostic software, GUI, abstraction, variables, data types and conditional statements).
TECH.8.2.12.E.CS1	Computational thinking and computer programming as tools used in design and engineering.

Target 1

Students will learn to create and assign values to variables.

Target 2

Students will use counter variables inside a 'for' loop.

Target 3

Students will distinguish between global and local variables, and use variables to track the score in a simple game.

Target 4

Students learn how to write and use if statements.

Learning Goal 3

Students will create a series of applications highlighting core concepts of programming

CRP.K-12.CRP2	Apply appropriate academic and technical skills.
CRP.K-12.CRP6	Demonstrate creativity and innovation.
CRP.K-12.CRP8	Utilize critical thinking to make sense of problems and persevere in solving them.
TECH.8.1.12.A	Technology Operations and Concepts: Students demonstrate a sound understanding of technology concepts, systems and operations.
TECH.8.1.12.B.2	Apply previous content knowledge by creating and piloting a digital learning game or tutorial.

TECH.8.1.12.B.CS1	Apply existing knowledge to generate new ideas, products, or processes.
TECH.8.1.12.B.CS2	Create original works as a means of personal or group expression.
TECH.8.1.12.C.1	Develop an innovative solution to a real world problem or issue in collaboration with peers and experts, and present ideas for feedback through social media or in an online community.
TECH.8.1.12.C.CS1	Interact, collaborate, and publish with peers, experts, or others by employing a variety of digital environments and media.
TECH.8.2.12.C	Design: The design process is a systematic approach to solving problems.
TECH.8.2.12.C.7	Use a design process to devise a technological product or system that addresses a global problem, provide research, identify trade-offs and constraints, and document the process through drawings that include data and materials.
TECH.8.2.12.D.1	Design and create a prototype to solve a real world problem using a design process, identify constraints addressed during the creation of the prototype, identify trade-offs made, and present the solution for peer review.
TECH.8.2.12.E	Computational Thinking: Programming: Computational thinking builds and enhances problem solving, allowing students to move beyond using knowledge to creating knowledge.
TECH.8.2.12.E.1	Demonstrate an understanding of the problem-solving capacity of computers in our world.
TECH.8.2.12.E.3	Use a programming language to solve problems or accomplish a task (e.g., robotic functions, website designs, applications, and games).
TECH.8.2.12.E.CS1	Computational thinking and computer programming as tools used in design and engineering.

Target 1

Students will create a useful graphical program, such as a painting program or scrolling image viewer

Target 2

Students will create simple animations or games using these concepts

Target 3

Students will explore further concepts such as while loops, arrays, and functions with parameters

Summative Assessment

- Quizzes & Tests
- Applied Projects

- Classroom Survey

Formative Assessment and Performance Opportunities

- Applied Activities/Projects
- Guided Practice
- Peer Review
- Reflective Discussion
- Teacher Observation
- Oral Questioning

Accommodations/Modifications

- Application problems for extra practice
- Scenarios for critical thinking
- Add additional features to the GooseCapture project

Unit Resources

Internet Resources

- Code.org's Computer Science Discoveries-Unit 3 curriculum
- Code.org's Computer Science Principles-Units 3 and 5 curriculum
    - *Some of the language in the Learning Goals, Targets and Essential Questions in these units borrows from or has been adapted from Code.org's curricula for its Computer Science Discoveries and Computer Science Principles courses, which are licensed via a Creative Commons license (Attribution-NonCommercial-ShareAlike 4.0 International-CC BY-NC-SA 4.0).*

Technology Software & Hardware

- Desktop computers
- Python programming language and IDE (Integrated Development Environment)

Textbooks (Online, pdf or print)

- Downey, Allen. *Think Python: How to Think Like a Computer Scientist* (2nd Edition). Needham, Massachusetts: Green Tea Press, 2015. http://www.thinkpython2.com.

Relevant Videos

- Code.org's video library

Interdisciplinary Connections

LA.RST.11-12.2	Determine the central ideas, themes, or conclusions of a text; summarize complex concepts, processes, or information presented in a text by paraphrasing them in simpler but still accurate terms.
LA.RST.11-12.3	Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks; analyze the specific results based on explanations in the text.
MA.K-12.3	Construct viable arguments and critique the reasoning of others.
MA.K-12.4	Model with mathematics.