Theme

Functions used as an element in creating a more complex function could be regarded not only as a collection of particular operations but also as an abstraction.

STEM Innovation Academy Unit 1

Subject: Introduction to Computer Science Unit Title: Building Abstractions with procedures Grade: 10	Teacher: Ms. Okoth Duration: 6 weeks

Summary of Units

In this unit, students will learn how to develop algorithms and implement them as procedures on various programming languages. Students will learn how to trace, compare and debug these algorithms. These skills are relevant throughout the iterative process of developing software. Students will learn how to identify the best algorithm and implements as procedures in the SNAP! Environment. Data structures like Lists, Queues, arrays, vectors, and Stacks will also be covered extensively throughout this unit

Standards/ Outcomes

9-12.CS.1 Describe ways in which abstractions hide the underlying implementation details of computing systems to simplify user experiences. (P4.1)

9-12.CS.3 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.

9-12.DA.10 Create data visualizations to help others better understand real-world phenomena. (P5.2)

9-12.DA.11 Refine computational models to better represent the relationships among different elements of data collected from a phenomenon or process. (P4.4, P6.3)

9-12.AP.12 Design algorithms to solve computational problems using a combination of original and existing algorithms. (P4.2, P5.1)

9-12.AP.13 Create more generalized computational solutions using collections instead of repeatedly using simple variables. (P4.1)

9-12.AP.15 Iteratively design and develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions. (P5.1, P5.2, P5.3)

9-12.AP.14 Justify the selection of specific control structures by identifying trade-offs associated with implementation, readability, and performance. (P5.2)

9-12.AP.16 Decompose problems into smaller sub-problems through systematic analysis, using constructs such as procedures, modules, and/or classes. (P3.2)

9-12.AP.17 Create computational artifacts using modular design. (P4.3, P5.2)
9-12.AP.18 Systematically design programs for broad audiences by incorporating feedback from users. (P1.1, P5.1)
9-12.AP.19 Explain the limitations of licenses that restrict use of computational artifacts when using resources such as libraries. (P7.3)
9-12.AP.20 Iteratively evaluate and refine a computational artifact to enhance its performance, reliability, usability, and accessibility. (P6.3)

9-12.AP.21 Design and develop computational artifacts working in team roles using collaborative tools. (P2.4)

9-12.AP.22 Document decisions made during the design process using text, graphics, presentations, and/or demonstrations in the development of complex programs. (P7.2)

9-12.IC.23 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices. (P1.2, P3.1)

9-12.IC.24 Identify impacts of bias and equity deficit on design and implementation of computational artifacts and apply appropriate processes for evaluating issues of bias. (P1.2)

9-12.IC.25 Demonstrate ways a given algorithm applies to problems across disciplines. (P3.1)

Stage 1 - Desired Results

Essential Questions:

- What is an "algorithm"?
- How can we construct an algorithm for performing simple tasks? (Days 1 - 3)
- What is SNAP! And how can we create small programs? (Days 4 - 6)
- Why are computer languages written in specialized languages (Days 5 -6)
- What are "blocks," "scripts," "sprites," and "the stage" in SNAP! (Days 7-8)
- How do we create a program with sprites, variables, cloning, listener events, and an infinite loop? (Days 9-10)
- How do we construct Algorithms that draw simple shapes? (Days 11-13))
- How do we animate SNAP! Sprites using costumes changes and movement? And How do we Trigger action in other Sprites using broadcasts (Days 14-16)
- How do we create an animated movie, play, nursery, rhyme, or other scene using the basic programming tools we learned? (Days 17-19)
- How do we use loops in SNAP!?(Days 20-21)
- How so we use nested loops? (Days 22-23)
- How do we ask for and receive user input in a SNAP! Program? How do we use simple conditionals to alter control flow in a SNAP! Program (Days 24-25)
- How do we use variables to track values throughout a program? (Day 26)
- How do we define, evaluate and utilize Boolean expressions and operators? (Days 27)
- How do we combine loops with conditionals to create models with repeated but conditional

behavior? (Day 28)
- How do we practice good style and conventions to create readable and maintainable code for a Pong video game? (29-30)

Unit pre-assessment:
- Lab 0.4 - Getting to Know You
- Lab 1.1 - Welcome to SNAP!
- Lab 1.2 - SNAP! Scavenger Hunt
- Lab 1.3 - Squares and Triangles and Stars, Oh, My!
- Lab 1.4 - Sprites in Action
- Unit 1 Quiz: SNAP! Basics
- Lab 2.1 - Squares and Triangles Redux
- Lab 2.2 - Another in the Wall
- Lab 2.3 - What Shape is that?
- Lab 2.4 - Guessing Game
- Lab 2.5 - Triangle of All Kinds
- Lab 2.6 - What goes Up…
- Unit 2 Quiz - Loops

Presentations:
- Product Iteration Process presentation
- Product and their implementation of 'The Six Professional Skills'
- Product simplicity and elegance

Stage 2 - Assessment Evidence

Performance Tasks:
- Define what is an "algorithm"
- Construct algorithms for performing simple tasks
- Complete levels in the game LightBot 2.0
- Complete small programs in SNAP with guidance
- Explain why computer programs are written in specialized languages
- Create a simple "program" in SNAP to describe themselves
- Define and identify "blocks," "scripts," "sprites," and "the stage" in SNAP.
- Write simple SNAP programs
- Describe what simple SNAP programs do without executing the code
- Name the categories of blocks in SNAP and describe what the blocks in each category do
- Describe the function of several common SNAP blocks (see lab for specific blocks)
- Use common blocks to build simple SNAP programs (see lab for specific blocks)
- Trace and debug through code
- Construct simple algorithms to draw shapes
- Convert algorithms into SNAP programs
- Write down the instructions you would give to a sprite to make it draw a triangle
- Create a SNAP program that draws some simple shapes on the stage
- Animate SNAP sprites using costume changes and movement
- Trigger action in other sprites using broadcasts
- Use costumes and movement to create simple SNAP animations

- Apply basic programming and SNAP skills to create an animated movie, play, nursery rhyme, or other scene
- Practice good debugging skills to correct issues as they arise while programming
- Project: Students will use SNAP basics to implement an animated version of a movie, play, nursery rhyme, or other scene.

Authentic experiences

- Lab: Getting to know you
- Project: Animated Storytelling
- Project: Create a Pong Game
- Project: Platform Game
- Project: Hangman

Extensions (Tier I)

- Product Iteration Presentation
- Accelerated SNAP! Route
- Game Programming Project
- AI Search algorithms
- More Programming Challenges

Differentiation for (Tiers II and III)

- Selective Grouping
- Small Group/Individual Instructions
- including TEALS volunteer
- Choice of format for Iteration Presentation
- Extended Time

Stage 3- Learning plan

Unit I Introduction to Computer Science (Beauty & Joy of Computing) Digital Content:
https://tealsk12.gitbook.io/intro-cs/ (Log in Required)

Vocabulary

- Abstraction
- Algorithms
- Brief first search
- Depth first search
- Procedures
- Higher order Procedures
- Functions
- Global
- Instance
- Binary Trees
- Values
- Types
- Propositional Logic

Resources

- Materials
- Tools
- Media
- Text
- Utilize google classroom Protocols
- Array
- List
- Set
- Queue
- Concurrent
- Prototype
- Modeling
  Design

Expert/field experiences

- Have Computer Scientists and Programmers from various industries speak
- TEALS Mentorship

Literature Connection and Research

- Iteration Research Presentation EQ #13, #8, EQ #15, EQ #16
- Project Presentations EQ #13, #15, EQ #16

APPENDIX

Students will know...	Students will be able to do…
• Various algorithms throughout their projects • Procedures that include functions that lead to higher order functions and objects • Create loops and logical properties that make up a computer program • Sprites and animations important for applications and games • Concurrent and Event Based Programming • Game Programming and the game mechanics and design behind various genres • State Machines • How to implement a graph algorithm • Propositional logic	• Various programs from simple to more complex applications and games • Present complex programs into easier understandable chunks • Troubleshoot and breakdown a program to find bugs and inefficiencies • Apply various software design principles; OOP, functional programming and understand which design process is best for their project