# Independent Study Project

## Enduring Understanding

The act of creating a prototype, identifying areas for improvement, and then refining your work to create a new prototype is a powerful process for developing an effective end product.

## Purpose

To create something that engages you and that you would be proud to share to a public audience.

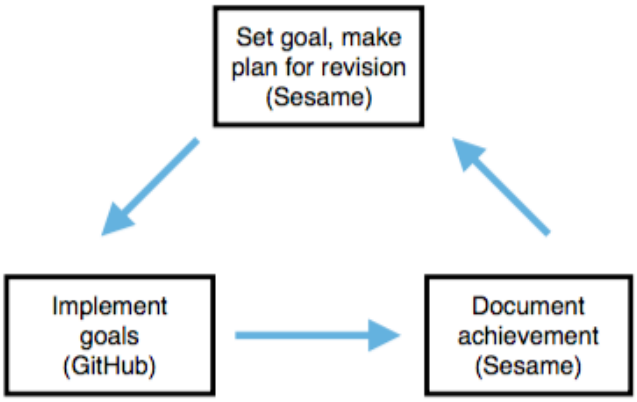
Along the way, you will build a solid working knowledge of programming syntax, structures, vocabulary, and source code management.

## Evaluation

You will be evaluated based on your ability to provide regular evidence that you have met curriculum expectations in this course.

Use GitHub and Sesame to provide evidence of what you have created and learned.

The more evidence you can provide, the better. Remember to apply this process:




Curriculum expectations have been sorted into categories. You will be assigned an overall grade in each category. I will provide interim updates on your grade in each category at two-week intervals until the end of the course.

## Knowledge

**B1.1** use correct terminology to describe programming concepts;
**B1.2** describe the types of data that computers can process and store (e.g., numbers, text);
**B1.3** explain the difference between constants and variables used in programming;
**B2.6** explain the difference between syntax, logic, and run-time errors;

## Inquiry

**B1.4** determine the expressions and instructions to use in a programming statement, taking into account the order of operations (e.g., precedence of arithmetic operators, assignment operators, and relational operators);
**B1.5** identify situations in which decision and looping structures are required;
**B2.1** use a visual problem-solving model (e.g., IPO [Input, Process, Output] chart; HIPO [Hierarchy plus Input, Process, Output] chart and diagram; flow chart; storyboard) to plan the content of a program;
**B3.3** use a tracing technique to understand program flow and to identify and correct logic and run-time errors in a computer program;
**B3.4** demonstrate the ability to validate a computer program using test cases.

## Communication

**B3.1** write clear and maintainable code using proper programming standards (e.g., indentation; naming conventions for constants, variables, and expressions);
**B3.2** write clear and maintainable internal documentation to a specific set of standards (e.g., program header: author, revision date, program name, program description; table of variable names and descriptions);

## Application

**B1.6** describe the function of Boolean operators (e.g., AND, OR, NOT), comparison operators (i.e., equal to, not equal to, greater than, less than, greater than or equal to, less than or equal to), and arithmetic operators (e.g., addition, subtraction, multiplication, division, exponentiation, parentheses), and use them correctly in programming.
**B2.2** use variables, expressions, and assignment statements to store and manipulate numbers and text in a program (e.g., in a quiz program, in a unit conversion program);
**B2.3** write keyboard input and screen output statements that conform to program specifications;
**B2.4** write a program that includes a decision structure for two or more choices (e.g., guessing game, rock-paper-scissors game, multiple-choice quiz, trivia game);
**B2.5** write programs that use looping structures effectively (e.g., simple animation, simple board games, coin toss);