Programming Module

Unit Overview:
This unit will fall in line with Code.org’s CS Principles curriculum for a CS1 class. It will take approximately 6-8 weeks of class time. The unit will introduce students to the Python programming language. The goal of the unit will be to familiarize students with the fundamental skills necessary to create working Python programs to solve problems. As students gain skills, they will learn to apply aspects of the language to make programs more efficient. By the end of the unit, students will begin to see where they could use programming to aid in other content areas.

Unit Outcome:
Understand how to create useful input/output programs utilizing the python programming language to aid student learning.

Possible expansions to unit outcomes:
- To understand why Python is a useful scripting language for developers.
- To learn how to design and program Python applications.
- To learn how to use lists, tuples, and dictionaries in Python programs.
- To learn how to identify Python object types.
- To learn how to use indexing and slicing to access data in Python programs.
- To define the structure and components of a Python program.
- To learn how to write loops and decision statements in Python.
- To learn how to write functions and pass arguments in Python.
- To learn how to build and package Python modules for reusability.
- To learn how to read and write files in Python.
- To learn how to design object-oriented programs with Python classes.
- To learn how to use class inheritance in Python for reusability.
- To learn how to use exception handling in Python applications for error handling.

Intermediate Outcome 1: Calculations
★ The first portion of the unit will introduce some basic information about the structure of Python and how to read and write programs. Students should have some background knowledge on coding from working with Code.org materials, but we will review key ideas such as algorithms, input/output, strings, ints and the overall program design process. We will also walk through the installation of IDLE (or whatever site we use). The unit will also discuss syntax and runtime errors and how to handle them. Finally we will start to look at how to utilize the input and print functions, set up variables and complete tasks that require designing programs that complete calculations.

★ Learning Activities-
Introduce the concepts of pseudocode and flowcharts to design/plan code before actually writing.

Students use a Google Doc to create a DIY project or recipe that has multiple steps. The project has to have at least 5 steps. The idea is to tie these steps that need to be done correctly to coding. If the steps aren’t completed or done in the correct order, the end result is not correct.

Lisa Olivieri has great POGIL resources that fit exactly what we envision our learning needs to be.

- Introduction to Python
- Input
- Calculations
- Output

Assessment will consist of a mixture of written questions that would be done on pencil and paper as well as the expectation to produce a program on a computer.

Sample problems-
- Explain what an algorithm is:
  - tax = 500
    - tax = 25 + tax
    - What would tax be in print(tax)
- Create a program to solve the following: If a person drinks 64 OZ of water in a day, how many cups do they drink in a year?

Intermediate Outcome 2: Booleans/Conditions/Selection.

The second topic will cover conditions and logical operators. Examples of this would include “>”, “<” and “=” as well as “and”, “or” and “not”. Utilizing these conditionals students can get more complex in their creation of programs and calculations. This will also go over selection statements including if-elif-and else for programs to decide a course of action by determining which conditions are true.

Assessment will again be a combination of written information along with a demonstration of a question answered via creating a program on a computer.

Learning Activities-
- Lisa Olivieri has great POGIL resources that fit exactly what we envision our learning needs to be.
  - Boolean Expressions
  - IF/THEN Statements
  - Nested If/Else Statements
Paired programming activity- Give students a problem with several if/then statements. One partner writes code, the other comments/observes. Switch after a given amount of time (10 min?)

Utilize a flowchart making program (such as Lucid Charts). Create a decision structure for a game or activity utilizing the program. Here is an example.

★ Sample problems-

- Assuming n=3, which of these is true?
  - (n > 6)
  - (n < 4) and (n > 1)
  - not (n = 3)
  - (n < 2) or (n < 1)

- Give a problem where students would calculate overtime pay using a given set of hours worked and wages earned.

Intermediate Outcome 3: Looping

★ The third topic will cover an introduction to looping (iteration) using Python. Looping is an extremely useful tool to use for any computer programmer in order to make programs work in an intended way. First, students will learn how to use repetition when you know how many times it will repeat (for loop) which will also help students to learn how to manage a loop index in Python. Then, students will learn how to deal with repetition when you don’t know how many times it will repeat (while loop).

★ Learning Activities -

- Lisa Olivieri has great POGIL resources that fit exactly what we envision our learning needs to be.
  - WHILE Loops
  - FOR Loops
  - NESTED Loops

- Tracing activities (given a few lines of code, the learner has to trace the order in which those lines are executed) are great for solidifying understanding of loops.

★ Assessment will consist of a mixture of written questions that would be done on pencil and paper as well as the expectation to produce a program on a computer.

★ Sample problems-

- What is the difference between range(10), range(0, 10), and range(0, 10, 1) in a for loop?

- Write a function `print_triangular_numbers(n)` that prints out the first n triangular numbers. A call to `print_triangular_numbers(5)` would produce the following output:
Write a short program that prints the numbers 1 to 10 using a for loop. Then write an equivalent program that prints the numbers 1 to 10 using a while loop.

Intermediate Outcome 4: Functions
★ The final topic of this module will be in creating functions, with an emphasis on the keywords `def` and `return`. The goal of this topic will be to help students create programs using function notation instead of the previously used input/output model. Although they will not get to see all the usefulness of creating programs in this way, it will help students who plan on taking more programming classes in the future.

★ Learning Activities -
  ○ Lisa Olivieri has great POGIL resources that fit exactly what we envision our learning needs to be.
    ■ Predefined Functions
    ■ void Functions
    ■ Value Returning Functions
  ○ Refactoring activities (given a working piece of code, the learner has to modify it in some way without changing its output) would be a great way to take old code and turn them into functions as some formative assessment.

★ Assessment will consist of a mixture of written questions that would be done on pencil and paper as well as the expectation to produce a program on a computer.

★ Sample problems-
  ○ When does the code in a function execute: when the function is defined or when the function is called?
  ○ What is a return value? Can a return value be part of an expression?
  ○ Fill in the body of the function definition for `cat_n_times` so that it will print the string, s, n times:

```python
def cat_n_times(s, n):
    <fill in your code here>
```