Chapter 1: Students should be able to:

- ☐ Evaluate compound expressions using rules of precedence and order of operations.
- ☐ Assign variables with allowed names and understand how to rebind that variable to new or different values.
- ☐ Utilize assigning multiple variables at the same time and understand when it might be useful to do so.
- ☐ Update variables using shorthand syntax. (`A += 1`)
- ☐ Define simple functions with inputs and outputs
- ☐ Import and use the `math` library for mathematical functions.
- ☐ Distinguish between and create Python objects of `int`, `float`, `bool`, and `string` classes.
- ☐ Identify what operations are viable on different types of basic objects (eg. You can add both floats and integers).
- ☐ Identify the resulting object type after an operation is performed (eg. Adding an `int` to a `float` results in a `float`).
- ☐ Convert between object types and identify situations in which the interpreter will automatically try to convert object types.
- ☐ Use `input` to get information from a user and understand what variable type is returned.

Chapter 2: Students should be able to:

- ☐ Construct program flow controls through the use of `if`, `elif`, and `else` statements with appropriate syntax.
- ☐ Parse complicated `if, elif, else` conditionals to decide what the output of a script might be.
- ☐ Evaluate expressions utilizing the logical operators `or` and `and`.
- ☐ Construct `while` loops with appropriate conditionals and which also terminate (no infinite loops!).
- ☐ Understand how nested loops behave and describe the output of each iteration of a set of nested loops.
- ☐ Understand what a predicate function is and be able to both understand and write one.
- ☐ Construct `for` loops with correct syntax over appropriate sequences.
- ☐ Identify situations where a `for` loop or a `while` loop might be more appropriate.
- ☐ Utilize the `range` function appropriately to construct ranges over desired intervals with valid step sizes.

Chapter 3: Students should be able to:

- ☐ Import necessary classes from the PGL library (`GWindow, GRect`, etc.
- ☐ Call and utilize common methods defined for a particular object class.
- ☐ Create a `GWindow` object with the desired dimensions.
- ☐ Create `GRect`, `GOval` objects with the desired dimensions and placed at the desired location on the window.
- ☐ Control the color and fill of any `GFillableObject`.
- ☐ Create `GLabel` objects with a desired font and placed in a desired location on the screen.
- ☐ Decompose larger problems into smaller, simpler problems which can be tackled one at a time.

Chapter 4: Students should be able to:

- ☐ Define a syntactically correct simple function.
- ☐ Understand and describe the difference between a function definition and a function call.
- ☐ Understand and describe the difference between parameters and function arguments.
- ☐ Utilize `return` statements in correct places in their code to return the desired value(s) *at the desired time* (and not earlier!).
- ☐ Call a function utilizing keyword arguments.
- ☐ Define a function utilizing default values for a formal parameter.
- ☐ Identify what variables are defined within a particular scope and what values they possess.
- ☐ Write an appropriate doc-string for a function, including a description of the function, what assumptions are made about inputs, and what guarantees the program makes about outputs that it returns.
- ☐ Import and use functions from Python's built-in `random` library.
- ☐ Write functions in a separate file and `import` them into a desired program.

Chapter 5: Students should be able to:

- ☐ Use functions as first class variables, assigning them to variable to be later used or returned by another function.
- ☐ Add event listeners to list for mouse events within a PGL graphics window.
- ☐ Define appropriate call-back functions to be called upon receiving an event.
- ☐ Use the `GState` object to share information between call-back functions *when necessary*.
- ☐ Retrieve what graphical objects (if any) are at a particular location on the graphics window.
- ☐ Create either interval or one-time timers which call a call-back function with some specific timing.
- ☐ Create a `GArc` object with desired dimensions and starting and stopping points at the desired location on the graphics window.
- ☐ Create `GPolygon` objects, with properly placed vertices, and desired locations in the graphics window.
- ☐ Create a `GCompound` object and add other graphical elements to that object and desired locations.

Chapter 6: Students should be able to:

- ☐ Describe the different between a number, and a representation of a number.
- ☐ Describe simple numbers in either decimal, binary, or hexadecimal representation.
- ☐ Describe why a computer's binary floating-point math sometimes gives slightly different results than our standard base-10 mathematical operations.
- ☐ Explain how Python represents characters internally as integers, according to an encoding scheme called Unicode, and how to convert back and forth between a character and its corresponding integer value.
- ☐ Define `str` objects and know what operations can (and can't) be done on strings.
- ☐ Access individual elements of a string through indexing.
- ☐ Slice strings to extract desired pieces with a starting point, a stopping point, and a stride size.
- ☐ Iterate through the elements of a string.
- ☐ Grow strings through concatenation.
- ☐ Use built-in common string methods to manipulate or search strings.
- ☐ Format strings nicely using `.format()` or f-strings.
- ☐ Use the `english.py` library as a source of valid English words.