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`, `string` and `list` types.
- ☐ Identify what operations are viable on different types of basic objects (eg. You can use `+` for floats, integers, strings and lists).
- ☐ Concatenate items to a string or list
- ☐ Select specific items from a string or list
- ☐ Determine the length of a sequence
- ☐ 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.
- ☐ Print variables or other text to the screen

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:

- ☐ Describe what an algorithm is
- ☐ Utilize the simple `english.py` library for certain word-related problems.
- ☐ Write a simple test function to test the correctness or output of another function.

Chapter 4: Students should be able to:

- ☐ Import necessary classes from the PGL library (`GWindow, GRect`, etc.)
- ☐ Call and utilize common methods defined for a particular PGL 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 5: 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 inputs are required and their types, and what, if anything, the function returns.
- ☐ Import and use functions from other libraries, in particular Python's built-in `math` or `random` library.
- ☐ Write functions in a separate file and `import` them into a desired program.

Chapter 6: Students should be able to:

- ☐ Use functions as first class objects, assigning them to variable to be later used or returned by another function.
- ☐ Add event listeners to listen for mouse events within a PGL graphics window.
- ☐ Define appropriate call-back functions to be called upon receiving an event.
- ☐ Use the `GWindow` object to share information between call-back functions when necessary.
- ☐ ~~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, at desired locations in the graphics window.~~
- ☐ ~~Create a `GCompound` object and add other graphical elements to that object before placing at a desired location.~~