Chapter 6: Students should be able to:

O1: □ Describe the different between a number, and a representation of a number.

O2: □ Describe simple numbers in either decimal, binary, or hexadecimal representation.

O3: □ Describe why a computer's binary floating-point math sometimes gives slightly different results than our standard base-10 mathematical operations.

O4: □ 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.

O5: □ Define `str` objects and know what operations can (and can't) be done on strings.

O6: □ Access individual elements of a string through indexing.

O7: □ Slice strings to extract desired pieces with a starting point, a stopping point, and a stride size.

O8: □ Iterate through the elements of a string.

O9: □ Grow strings through concatenation.

O10: □ Use built-in common string methods to manipulate or search strings.

O11: □ Format strings nicely using f-strings and format specs.

O12: □ Use a provided `english.py` library as a source of valid English words.

Chapter 7: Students should be able to:

O13: □ Create a Python list (array) with proper, valid elements inside.

O14: □ Concatenate, remove entries, index, slice, and loop over lists.

O15: □ Create lists compactly using the list comprehension syntax.

O16: □ Open a text file to be read and looped over, performing desired operations.

O17: □ Open a text file to be written and add desired content.

O18: □ Use a list for tabulation, incrementing an index when some desired event occurs.

O19: □ Utilize a try-except statement to make it possible for a program to smoothly handle an exception or error condition.

O20: □ Create multi-dimensional arrays and access specific elements within a multi-dimensional array.

O21: □ Use PGL's `GImage` class and associated methods to convert images to multi-dimensional arrays of pixel values.

O22: □ Manipulate the colors of pixels of a `GImage`.

Chapter 8: Students should be able to:

O23: □ Determine which of two programs (each of which would run in some polynomial time) would run quicker for a given N.

Chapter 9: Students should be able to:

O24: □ Create a new simple class from scratch with an appropriate constructor that defines new attributes for the class.

O25: □ Write getter and setter methods to retrieve or manipulate class attributes.

O26: □ Define a dunder method so that instances of the class are printed nicely to the screen.

O27: □ Create objects which are instances of a custom defined class.

O28: □ Use receiver syntax to call class methods on an instance.

Chapter 10: Students should be able to:

O29: □ Add, remove, and change key/value pairs in an existing dictionary.

O30: □ Access or lookup values corresponding to different keys in a dictionary.

O31: □ Iterate through a dictionary.

O32: □ Create a Python set object with a non-zero number of valid elements.

O33: □ Utilize built-in methods for set objects to compare or check membership.

Chapter 11: Students should be able to:

O34: □ Choose appropriate data structures for a given type of information

O35: □ Comfortably nest data structures within others to model and store more complicated data

O36: □ Retrieve or select specific information from within a nested data structure

O37: □ Read JSON data from an external file into native Python data types

O38: □ Use a provided ADT, given its documentation

O39: □ Utilize a data structure to make a program or piece of a program data-driven, wherein the provided data structure dictates the flow of the program.

O40: □ Convert a data structure from an *external format* as it is written in some text file to an *internal format* using some hierachical combination of custom or built-in objects.