ENGR 102 - Midterm Learning Objectives

Given a piece of Python code, be prepared to answer questions such as:

- what will the code output,
- what values are stored in memory, and of what data type
- what type a variable is,
- where there is an error or bug

Be prepared to answer questions regarding *definitions* or *behavior*, write *test cases*, *list steps* in a computation, *outline a design* and *write short programs* on paper. Expect to combine several covered topics into problem solutions.

In reviewing for the midterm, be familiar with all the topics we have covered. You should be able to:

Introduction to Programming

- Define the terms IDE, Compiler, and Interpreter
- Use the print statement in Python
- Define the purpose of, and utilize in a program, the import function in Python (e.g., from math import *)
- Use simple mathematical operations in Python (+, -, *, /, //, %, **, sqrt, cos, sin, tan, log, log10, exp)
- Use comments in a Python program

Sequential Steps, Variables, Assignments and Data Types

- Utilize variables in Python programming to store data
- Apply Python naming rules to variables
- Identify necessary programming variables from a given problem statement
- Assign or reassign values to variables
- Use the special assignment operators (+=, -=, *=, /=)
- Write a sequence of steps to successfully complete a complex task
- Follow a sequence of steps to understand the complex task that was performed
- Identify the integer, floating-point number, Boolean and string data types in Python
- Know common usages of these data types in programming
- Create, read and use each data type within a Python program
- Use the escape character (\) in Python to define special characters in strings
- Convert between data types in Python, and know when an error will occur
- Find the value of a calculated expression during data type conversions

Input, Output and Calling Modules and Functions

- Format output in good form using the print command
- Concatenate strings using the + operator
- Modify the print item-separator and line-end commands (sep = "", end = "")
- Use the input command to read information from the user
- Provide a useful prompt to the user with the print command
- Use input information to perform calculations within Python
- Use the newline command (\n) in Python where appropriate
- Define the terms function call and return
- Explain and give examples of the benefits of functions
- Import modules to a program following good programming practice (i.e., don't use import *)
- Call functions from modules correctly (e.g., math.cos())
- Use a function with or without parameters
- Use a function with or without return variable
- Utilize any function if provided with a function definition or description

(Continued, next page)

Conditionals and Boolean Expressions

- Identify a linear and a branching process from an ordering of steps or a flow chart
- Create and solve Boolean expressions using relational operators, following correct order of operations
- Create Boolean expressions in Python
- Use if, elif and else statements to correctly branch a Python program
- Identify correct instances to use if-only, if-else, and if-elif-else forms of branching in a program.
- Use nested conditional statements

Creating and Testing Programs & Basic Debugging

- Utilize comments in your program to section code, define and clarify variables or computations, refer to sources and improve readability
- Define incremental coding, pyramid- and arch-style software construction, and the terms typical, edge and corner as they relate to program testing
- List major steps of solving a problem
- Structure a program by using comments from the problem-solving step list
- Write test cases prior to writing programming code that will incrementally verify your program
- Write tests that demonstrate your program calculates correct results for a wide variety of cases, including unlikely scenarios
- Create and test code incrementally
- Describe a proper debugging process
- Describe the use of an interactive debugger tool in a programming IDE

Loops and Iteration

- Identify a repetition process from an ordering of steps or a flow chart
- Create a while loop in Python, and correctly form the conditional statement
- Recognize an infinite loop, and how to correct one
- Create a for loop in Python, and correctly define a range for iteration
- Identify values of variables and iterators for each iteration through a loop
- Use nested looping statements

Lists of Data and List Operations

- Create and identify lists in Python
- Assign or reference a single element within a list
- Find the length of a list using the len() command
- Loop through the elements of a list where the iterator is the index value through a range
- Loop through the elements of a list where the iterator is the value of each list element
- Create and index lists of lists
- Use the list operations of append, insert, and concatenation
- Use within a program the functions: len(), max(), min(), sum(), .append(), .insert(), .sort(), .index(), .pop()
- Slice a list to print, delete, or replace portions of a list or create a new list
- Use the shortcut methods of slicing to refer to the start or end of a list (e.g., [:b], [a:], [:])
- Slice a string to print or define a new string
- Recognize limitations of slicing when used on string data types and when errors will occur

(note: the learning objectives list on the course slides regarding tuples and dictionaries will not be covered on exam 1)