AAI/CPE/EE 551.A – Engineering Programming: Python

Exam 1 Study Guide

The midterm exam covering Chapters 1, 2, 3, 4, 5, 6, 7, 8, and 9 in the textbook will consist of a written exam containing multiple choice, true/false, fill in the blank, and short answer questions. Students will take their exam during their class session on Friday, October 25th. You are allowed one side of an 8.5"x11" or A4 sized piece of paper for handwritten notes. You will have 1 hour 30 minutes to complete this exam.

You should review your textbook, your notes, the lecture videos, the quizzes, class code examples, posted tutorials, and previous lab assignments to prepare for this exam.

You should be familiar with the structure of a basic Python program and its main components, such as declarations, blocks, assignments, etc. You should also be familiar with the following topics:

- Declaring, initializing, and using variables
- General syntax rules, such as proper indentation and use of :
- Using `print()` for displaying output
- Using `input()` for reading user input, and properly casting the input as necessary
- How to perform arithmetic calculations, including order of operations, as well as use of compound operators (i.e., +=, /=, etc.)
- The fundamental data types, such as integral types and floating-point types, including working with characters
- Python style comments using `#` and docstrings
- Data conversions and type casting
- Generating random numbers using the functions in the `random` module such as `random()`, `randint()`, and `randrange()`
- Branch statements such as `if`, `elif`, and `else` statements, especially in constructing branching statements using equality (e.g., ==, `!=`) and relational operators (e.g., <, <=, >, >=).
    - Use `and`, `or`, and `not` operators for compound Boolean expressions
    - Use conditional expressions in place of `if-else` branches
- Bitwise operations (`&`, `|`, `^`)
- Loops (e.g., `while` and `for` loops) to accomplish iterations for some task
- Use of the different versions of the `range()` function in conjunction with `for` loops to control iteration
- Proper use of `break` and `continue` statements when working with loops
- Boolean expressions, including the `boolean` data type, and how to evaluate them
- Declaring and initializing string variables for accessing, searching, comparing, and modifying using appropriate functions
- Declaring both a one-dimensional and two-dimensional `List`
- Initializing a `List` *at the same time* it is declared in addition to using a `for` loop
    - When working with `Lists`, data can be used to populate a `List` either manually (such as user input) or automatically (such as random numbers).

- Processing a 1D- or 2D-`List` (i.e., using a `for` loop)
- List comprehensions for producing new lists using other data
- Generator comprehensions for producing iterators using other data
- The basics of Jupyter Notebooks and how they store data and execute code
- Defining a function and specifying its parameter list, potentially containing multiple parameters
- Assigning default values to function parameters
- Using keyword arguments when calling a function
- Understanding and compensating for the difference between locally and globally scoped variables
- The Python `math` module and its functions and constants
- Separating a program into multiple module files and being able to import modules in other modules
- Creating and using lambda functions to perform simple calculations
- Using `pytest` module to create tests for programmer defined functions
- Using marking to provide additional configuration or information regarding a test
- Opening and closing a user specified file
- Checking if a user specified a file exists
- Understanding the difference between absolute and relative path locations
- Reading data from or writing data to a file
- Creating a new file
- Parsing and processing data that has been read from a file using functions such as `split()`, `strip()`, `find()`, and `[:]`
- Handling exceptions using `try/except/else/finally` statements
- How to `raise` an exception in the event something goes wrong in a program
- Accessing information from an exception and printing it to the user
- Using a `Tuple` to store and access data using its functions and operators
- Using a `Dictionary` to add, store, remove, and access data using its functions and operators
- Using views to access data in a `Dictionary`
- Using a `set` to add, store, remove, and access data using its functions and operators
- Using `set` specific functions such as union, intersection, difference, and symmetric difference
- Determining if a `set` is a subset or superset of another `set`
- Using pickling to store and retrieve data as binary streams
- The characteristics of programming languages
- Using the `pandas` module for reading in, performing basic processing, and writing out data