7001 Python Programming Exercises

The following are coding exercises to test your understanding of basic Python functionality that you will encounter in the MS program. Most of the coding questions involve writing a function to get the job done and passing parameters to the function. Be aware that Python is NOT a strongly typed language which means the type of parameters are not checked by the compiler.

However, Python 3.5 and above have a way to document s and this information may be used by tools to help when you incorrectly pass an integer when your code expects a string.

In these exercises, we'll use the following format to specify what your function should look like, what data types are expected to be passed in and what kind of return value is expected.

The function below takes and returns a string and is annotated as follows:

```
def greeting(name: str) -> str:
    return 'Hello ' + name
```

In the function `greeting`, the argument `name` is expected to be of type `str` and the return type `str`.

Note that Python compilers will not report type errors (like Java, and C++ do). These type annotations are for documentation purposes and are used to document your intent.

The types used here are: int, float, bool, List, Tuple, Dict, None, Any (expect any type) , Number (int or float)

If you were asked to take this course as part of your admission, you should plan to submit the results of your coding explorations. For each assignment, capture the code and the output and submit as a PDF file. When you prepare your submissions, use COURIER font which is the standard for code examples. Courier font provides uniform spacing is makes it easy to read. Happy coding!

Assignment 2 Files

1. Create a 10 line text file with some text of your choice. Write a Python to read first n lines of the file and return the number of alphabetic (a-z A-Z) characters you read in for the n lines.

: def read_count (filename: String, n: int) -> int:

2. Write a Python to write each element of a list to a file, line by line. Pass in the list and filename as parameters.

: def list_to_file (mylist: List, filename: String) -> None:

Test with the list: [2,4,6,8,10,33] and with the list ['foo', 'bar', 'baz', 'bop']

3. Create two text files. Write a function to append the contents of the first file to the second file. Pass the two filenames as parameters. Display the contents of filename2, the appended file, after the function returns.

function: def append_file (filename1: String, filename2: String) -> None: