For this assignment:

- **You are expected to write all the code you use. You may not rely on functions found in Python libraries, or on function-like constructs supported by Python. <u>You will receive no credit for any function that violates this requirement.</u> Read below for details.**
- you **<u>MAY use</u>**
  - the Python function **len**.
  - **while** loops
  - **for** loops.
- **you <u>MAY NOT use</u>**
  - the ***in*** operator except with ***for*** loops. For example, you cannot use ***if num not in a_list.***
  - any other Python data structures, for example dictionaries,
  - Python built-in functions like *min, max*, *sum*, *sort,* or other list methods.
  - other Python functions like ***shuffle***, or functions in other Python modules that solve the problem for you
- **In addition to the 5 functions below, you may write other supporting functions as necessary.**

Download the following files to the same folder:

1. test_funcs.py - contains tests for the functions you write. I will use this code to test the code you submit.
2. testing.py - contains code used for testing. Used by *test_funcs*
3. time_this.py - you will use this to measure execution time for your functions
4. funcs.py - contains stubs for functions that you will complete as described below

- After you write the functions, run *test_funcs.py* to test your functions for correctness.
- **Analyze <u>each function</u> as described below. For each function, your analysis <u>must</u>:**
  - specify how the **size of the problem** is determined. For example, for a function that operates on a list, the size of the list may determine the size of the problem.
  - **Include a brief analysis** that shows how you **determined its complexity using big-O notation**. The analysis must reference your code, count operations, and clearly show how you arrived at your answer.
  - Include results of experiments that you run to measure the execution time of the function different problem sizes as detailed below:
    - For problems 1,2,3,4, run experiments for problem sizes of 2000, 4000, 8000, 16000, 20000.
    - For problem 5, run experiments for problem sizes of 2,4,8,16,20.
    - Report your results using a **two-column table showing problem size, and execution time**
    - Show that your experimental results are consistent with your analysis of the algorithmic complexity of the function. For example, if your analysis shows that your algorithm has complexity ***O(N)***, your experiments will show that

doubling the problem size doubles the execution time.

- **Upload the following to Canvas**:
  - **funcs.py**
  - One Word file containing your report of the analysis of each function.

1. ***def average(a_list):***
   - *returns the average value of a list of numbers*
2. ***def moving_average(a_list):***
   - *returns a list such that the n^{th} element of the list represents the average of the first n elements of* ***a_list****.*
   - *Example: if* ***a_list*** *= [3, 4, 14 ,15, 4], function returns the list [3, 3.5, 7, 9, 8]. The n^{th} element of the second list is the average of the first n elements of* ***a_list****.*
3. ***def is_unique(a_list):***
   - *returns True if no two numbers in* ***a_list*** *are the same, i.e.* ***a_list*** *contains no duplicate numbers. False otherwise.*
4. ***def diff(a_list, b_list):***
   - *returns a* ***list*** *containing all numbers that are present in either* ***a_list*** *OR* ***b_list****, but not in both. For example, diff([9,2,4,3,10,9,5], [9,4,2,3,5,9,11]) returns the list [10, 11]. The contents of the list need not be in any particular order.*
5. ***def four_letter_words(n) -***
   - *returns a* ***list*** *of all the four-letter words that can be constructed using the first* ***n*** *letters of the uppercase alphabet. For example,* ***four_letter_words(2)*** *returns all the four-letter words that can be formed using he first 2 letters of the uppercase alphabet, i.e. A and B. In this case, the function returns the list* ***['AAAA', 'AAAB', 'AABA', 'AABB', 'ABAA', 'ABAB', 'ABBA', 'ABBB', 'BAAA', 'BAAB', 'BABA', 'BABB', 'BBAA', 'BBAB', 'BBBA', 'BBBB'].*** *The contents of the list need not be in any particular order.*
6. ***Grading:***
   1. ***(105 points) Implementation, analysis, and experimental results - For each of the five functions, the implementation of the function, its analysis, and experimental results are worth 7 points each.***