Assignment #1: Python and NumPy Exercises
Due: Friday, March 5th, 2021 at 11:59 PM (EST)

Description
In this homework you will practice your coding skill in Python with NumPy package. The goal of this assignment is to help you get familiar with NumPy functions and write a program in Python. Furthermore, you will measure the timing differences between pure (or vanilla) python and NumPy when operating on arrays.

Instructions
In this homework assignment, you will need to write a program in Python that performs the 15 steps listed below. The steps have be broken down into 3 sets to highlight the learning objectives. For each step, you need to write **2 versions** of the code. One will use pure Python and the other will use the NumPy package that was introduced in class. You will also be asked to collect the run time information of the two versions of the program, analyze and compare their speed in your report.

For each of the versions, you will be asked to store the results in a particular variable. In order to differentiate between the two version, please use the following convention of appending "**_1**" to the pure Python version and "**_2**" to the NumPy version.

> Example: Create a zeros vector of size 10 and store it in variable **tmp**.
> Python answer: tmp_1 = [0 for i in range(10)]
> NumPy answer: tmp_2 = np.zeros(10)

As you complete each step, you will need to time how long it takes to complete each task in both the Python and NumPy version separately. An example of how to do this with the **time** package can be seen in the **example()** function provided in the provided Python file.

Notes:
- All steps are listed in assignment1.py.
- Fill in the code for each step in assignment1.py.
- You will also see an example time function in assignment1.py that is used to measure the CPU time to execute this block of code. Record the value, you will later use them to compare the speed of Pure Python and NumPy implementations.
- For all variables in pure Python implementation, extend the variable name with "_1" at the end (See example above).
- For all variables in NumPy implementation, extend the variable name with "_2" at the end (See example above).
- In the report, plot bar charts to compare the execution time of each block of code, comparing your pure Python and NumPy implementations.

Submission
Your submission will contain 1 python file named as '**assignment1.py**', and 1 report named '**report.pdf**'
- Zip file named via the following convention:
  - <SU-EMAIL>_<FIRST-Name>_AS1.zip
  - Ex. dprider_Daniel_AS1.zip
- Upload the zip file to blackboard before 11:59PM (EST Time) 03/05/2021
- In the report, plot the bar chart to compare the execution time of each block of codes for pure Python and NumPy implementation.

Assignment #1: Python and NumPy Exercises
Due: Friday, March 5th, 2021 at 11:59 PM (EST)

Steps:

The steps are provided both here and in the **assignment1.py** file provided.

I. Set #1: Manipulating Elements
  1. (5) Create a zeros array of size (3,5) and store it in variable **z**.
  2. (5) Set all the elements in first row of **z** to 7.
  3. (5) Set all the elements in second column of **z** to 9.
  4. (5) Set the element at (second row, third column) of **z** to 5.

II. Set #2: Creating Specialized Arrays
  5. (5) Create a vector of size 50 with values ranging from 50 to 99 and store it in variable **x**.
  6. (5) Create a 4x4 matrix with values ranging from 0 to 15 and store it in variable **y**.
  7. (10) Create a 5x5 array with 1 on the border and 0 inside and store it in variable **tmp**.

III. Set #3: Functions on Arrays
  8. (10) Generate a 50x100 array of integer between 0 and 5,000 and store it in variable **a**.
  9. (10) Generate a 100x200 array of integer between 0 and 20,000 and store it in variable **b**.
  10. (10) Multiply matrix **a** and **b** together (real matrix product) and store the result to variable **c**.
  11. (10) Normalize a 3x3 random matrix ((x-min)/(max-min)) and store the result to variable **d**.
  12. (5) Subtract the mean of each row of matrix **a**.
  13. (5) Subtract the mean of each column of matrix **b**.
  14. (5) Transpose matrix **c**, add 5 to all elements in matrix, and store to variable **e**.
  15. (5) Reshape matrix **e** into a 1d array, store it to variable **f**, and store the shape of **f** to variable **g**. Finally, print **g**, the shape of **f**.