Python Week 2

Due by Sunday at the end of Week 2 (11:59 pm Central)

1 Directions and Policies

1.1 Collaboration Policy

You are welcome to ask your instructor for help, but this project must be completed **individually**. It is expected that whatever work you submit to Gradescope is your own.

1.2 What to submit

You will submit one file to Gradescope with the filename `pweek2.py`. It is very important that you name your file exactly as described above.
The rest of this document will explain what should be in this file.

1.3 What is this assignment about?

You are going to write functions that perform some of the basic linear algebra calculations performed on matrices. Specifically, you will be working with the NumPy module in this assignment. If you haven't already taken a look at the NumPy documentation, I would highly recommend it.
You might also find this question and answer session on Stack Overflow helpful for some of the functions below.
In order to make it easier to program with indices, we will refer to the top row of a $m \times n$ matrix as row 0 and the bottom row as row $m-1$. The same convention will be done for columns. We follow this convention since `ndarray`'s in NumPy are 0-indexed.

2 Functions to write

Your `pweek2.py` file should contain the following functions.

2.1 The `SwapRows` function

Define a function `SwapRows(A,L)` where `A` is a NumPy array, `L` is a tuple with two elements, and we want to switch the two rows `L[0]` and `L[1]` of `A`. The function should return the resulting array.

2.2 The `MulRow` function

Define a function `MulRow(A,r,c)` where `A` is a NumPy array and `c` is the non-zero constant to multiply row `r` by. The function should return the resulting array.

2.3 The `AddMul` function

Define a function `AddMul(A,L,c)` where `A` is a NumPy array, `L` is a tuple with two elements, and row `L[1]` is added to `c` times row `L[0]` and the sum replaces row `L[1]`. The function should return the resulting array.

2.4 The `Pivot` function

Define a function `Pivot(A,L)` where `A` is a NumPy array and `L` is a tuple with two elements, which will pivot on the `L[0],L[1]` element of `A` if that element is non-zero, and return `0` if the element is zero. Recall that to **pivot** on an element means to make that element equal to 1 and every other element in that column equal to 0.

2.5 The `rref` function

Using the `Pivot` function from above, define a function `rref(A)` which will output a 2-tuple where the first element is the reduced row echelon form of `A` and the second element is the rank of `A`.
For example, the following lines of code

```
import numpy as np
A = np.array([1,2,3,4,5,6]).reshape(2,3)
print(rref(A))
```

should produce the output
`(array([[ 1, 0, -1], [ 0, 1, 2]]), 2)`
Recall that the **rank** of a matrix is the number of leading ones when put into reduced row echelon form. The `Pivot` function should NOT use any built-in functions of Python (or any of its modules) to calculate the rank of `A`. Instead, it should count the number of pivots.

2.6 The `PoolMatrix` function

You are going to define a function that **pools** a matrix. This technique is used to simplify a matrix of values. There are multiple ways to pool a matrix. In this problem, we are going to consider two types of pooling: max pooling and average pooling. See the two websites before continuing with this problem. It is also recommended that you do some manual calculations with pooling before you try to write the function described below.
Note that average pooling is just like max pooling except we take the average of all the elements in a filter rather than the maximum of all the elements in a filter.
Define a function `PoolMatrix(A,f,type=0)` that will take two required arguments `A` and `f`, and one optional argument `type`. The input `A` is a square matrix to pool inputted as a NumPy array, `f` represents a `f x f` filter size, and `type` will default to `0` to mean max pooling. If `type=1`, then this will mean average pooling.
We will assume the stride length is equal to 1. Once again as said above, we will further assume that `A` is a square matrix.

3 Examples

Below are some examples of what the output of these functions should be. In each example, we are using
`A = np.array([1,2,3,4],dtype=float).reshape(2,2)`
The results below would be the returned results, respectively, from the following commands:

```
SwapRows(A,[0,1])

MulRow(A,1,5)

AddMul(A,[0,1],-1/3)

Pivot(A,[1,1])

#SwapRows(A,[0,1])

[[3. 4.]
 [1. 2.]]

#MulRow(A,1,5)

[[ 1.  2.]
 [15. 20.]]

#AddMul(A,[0,1],-1/3)

[[1.         2.        ]
 [2.66666667 3.33333333]]

#Pivot(A,[1,1])

[[-0.5   0.  ]
 [ 0.75  1.  ]]
```

Here are some pooling examples:

```
A = np.arange(16).reshape(4,4)

> PoolMatrix(A,2,1)

[[ 2.5  3.5  4.5]
 [ 6.5  7.5  8.5]
 [10.5 11.5 12.5]]
```

```
> PoolMatrix(A,2)

[[ 5  6  7]
 [ 9 10 11]
 [13 14 15]]

#################
#Another example#
#################

B = [[0.79316344 0.94454549 0.34180841 0.92588053 0.81812942]
 [0.32217859 0.12929415 0.28896883 0.82325678 0.01963341]
 [0.41620186 0.03782866 0.51917636 0.52361364 0.2345459 ]
 [0.00918716 0.8243298  0.86251483 0.03936222 0.12105556]
 [0.34449926 0.62654229 0.11324716 0.90563128 0.54266996]]

> PoolMatrix(B,3)

[[0.94454549 0.94454549 0.92588053]
 [0.86251483 0.86251483 0.86251483]
 [0.86251483 0.90563128 0.90563128]]
```

3.1 IMPORTANT NOTE

You should test your code by printing and checking the results. However, you should suppress all print statements, both inside and outside the functions, when you're submitting to the autograder. If you get a "parsing" error when submitting to the autograder, that could be one of the issues.
If you are ever unsure of an autograder error, don't hesitate to ask your instructor.

4 How will the assignment be graded?

4.1 Autograder: 90 points

The autograder tests your script and grades it based on execution and sample inputs. The following tests will be run:

1. Did it receive a file named `pweek2.py` and nothing else? (5 points)
2. Does the Python interpreter accept the file `pweek2.py` as valid Python code? In other words, can the script be run without errors? (5 points)
3. Does the file `pweek2.py` contain definitions of each of the four functions described above? (5 points)

4. Sample tests will be run on the functions. (75 points)

4.2 Manual review: 10 points

Your code will be checked manually for readability, proper comments, and anything the autograder cannot check. A good rule of thumb is one comment should be used for every 4 to 5 lines of code. If someone (or yourself a few years from now!) reads this code, they should be able to read your comments to get a feel for what you are doing.

4.3 Autograder workarounds

If we see that your program does not do what was requested in this document, but the error was not detected in the automated testing, a deduction may be given at this point. The scores assigned by the autograder will not be changed during manual review unless we discover some kind of intentional wrongdoing (such as an attempt to circumvent or reverse-engineer the autograder's operation).

4.4 Final word

It is **really** important to test locally
As you write your project, test it locally on your own computer. Do not try to test it with only the autograder. Make sure your programs run on your own computer before uploading them to Gradescope. It is much harder to debug a broken program based solely on reports you get from the autograder compared to working with your local Python interpreter.