Python Programming Section 1832 & 1833

Deadline Nov 16, 2020 by 11:59 PM PST

Programming Assignment 4 (75 Points): More Fun With Functions & Python Lists

Assignment Description (minted - November 9, 2020):
The fourth assignment will use Python lists and the **def** keyword to create functions that work on $1D$ and $2D$ lists. In mathematics, a $1D$ list of numbers can be called a **vector**, a $2D$ list of numbers can be referred to as a **matrix**, assembled into a table of rows and columns, and a single number can be called a **scalar**. In Python, a vector can be created with a $1D$ list and a matrix can be created using $2D$ lists.

For this assignment, we will create three functions. One to compute the dot product between two $1D$ lists of numbers of any length, one to compute the transpose of a $2D$ list of lists (i.e. a matrix), and one for scalar matrix multiplication. All three functions return a value and are described below with their respective example usage.

Program 1: Dot Product Between Two Vectors (25pts)

The dot product between two vectors $\vec{a} \cdot \vec{b}$ where $\vec{a} = (a_1, a_2, \ldots, a_n)$ and $\vec{b} = (b_1, b_2, \ldots, b_n)$ is computed using the formula:

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i * b_i = a_1 * b_1 + a_2 * b_2 + \ldots + a_n * b_n$$

Write a function called **dot_product(a,b)** that returns the dot product as a scalar value (i.e. a single number). The dot product can be computed from two n length vectors $\vec{a}$ and $\vec{b}$. The original vectors $\vec{a}$ and $\vec{b}$ must not be modified. Use a $1D$ Python list to store each vector of numbers input. Only one value is returned.

Example usage of dot_product(...):

```
>>> a=[5,-8];b=[1,2]
>>> dot(a,b)
-11
>>> a = [1,2,3];b=[4,5,6]
>>> dot(a,b)
32
>>> a=[0,3,-7,1];b=[2,3,1,3]
>>> dot(a,b)
5
>>> a=[0,3,-7,1,3];b=[9,5,2,1,6]
>>> dot(a,b)
20
```

Program 2: The Transpose of a Matrix (2D List) (25pts)
The transpose of matrix A is a new matrix $A^t = transpose(A)$ where the rows of A are the columns of A^t and the columns of A are the rows of A^t. The matrix A does not need to be a square matrix. For example, the transpose of the matrix A defined below is:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \text{ transpose(A)} = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}$$

Write a function called **transpose** that will return a new matrix with the transpose matrix. The original matrix must not be modified. See the usage for examples of how **transpose** works. Use a $2D$ Python list as shown to store the input and output matrices.

Example usage of transpose:

```
>>> m5 =[[1, 2, 3, 4], [5, 6, 7, 8]]
>>> m6 = [[1, 2, 3, 4]]
>>> m7 = [[1, 2], [3, 4], [5, 6]]
>>> transpose(m5)
[[1, 5], [2, 6], [3, 7], [4, 8]]
>>> transpose(m6)
[[1], [2], [3], [4]]
>>> transpose(m7)
>>>[[1, 3, 5], [2, 4, 6]]
```

Program 3: Scalar Matrix (i.e. 2D list) Multiplication (25pts)
A matrix A can be multiplied by a scalar value v to create a new matrix $B = scalar\_multiply(v, A)$ where each element of A is multiplied by the scalar value v.

Given $v = 3$ and A as shown below, B can be computed as $B = scalar_multiply(v, A)$:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \text{ scalar\_multiply(v,A)} = \begin{bmatrix} 3 & 6 & 9 \\ 12 & 15 & 18 \\ 21 & 24 & 27 \\ 30 & 33 & 36 \end{bmatrix}$$

Write a function called **multiply** that returns a new matrix with product of matrix AxB. The original matrices (A and B) must not be modified. See the usage for examples of how **multiply** works. Use a 2D Python list as shown to store the matrices.
Example usage of multiply:

```
>>> M = [[1,2],[3,4]]
>>> scalar_multiply(5,M)
[[5, 10], [15, 20]]
>>> N = [[5,6],[7,8]]
>>> scalar_multiply(1,N)
[[5, 6], [7, 8]]
>>> A = [[1,2,3],[4,5,6],[7,8,9],[10,11,12]]
>>> scalar_multiply(2,A)
[[2, 4, 6], [8, 10, 12], [14, 16, 18], [20, 22, 24]]
```

Where to do the assignment
You can do this assignment on your own computer, or from the SMC Virtual labs via Citrix. In either case, ensure the code runs on Windows and in IDLE. Submit one **.py** file named **A04.py**. Do not use any other name or else points will be deducted.

Submitting the Assignment
Include your name, your student id, the assignment number, the submission date, and a program description in comments at the top of your files. Submit the assignment on Canvas (`https://online.smc.edu`) by **uploading your .py file** to the Assignment 4 entry as an attachment. Do not cut-and-paste your script into a text window. Do not hand in a screenshot of your program's output. Do not hand in a text file containing the output of your program. Do not save and turn in the interpreter session (e.g. the one with $>>>$).

Saving your work
Save your work often on a flash-drive or to the cloud (e.g., GoogleDrive, Microsoft OneDrive, Canvas, etc.). Always save a personal copy of your files (e.g. .py, etc.). Do not store files on the lab computers.

Reference:

1. dot product: https://en.wikipedia.org/wiki/Dot_product
2. transpose: https://en.wikipedia.org/wiki/Transpose
3. scalar_multiplication: https://en.wikipedia.org/wiki/Scalar_multiplication