Python Workshop 5 Course Work

Answer the questions given below and submit your work to the Canvas portal provided at the end of the workshop session.

Example 1

There is no special structure in Python for representing a two-dimensional table. Typically each row of a table is represented as a list, and the whole table is represented as a list of lists, its rows. For example,

```
1  2  3
4  5  6
```

can be defined as:

```
>>> m = [[1, 2, 3], [4, 5, 6]]
```

Actually, Python lets you split lists between lines, so you can write

```
>>> m =[[1, 2, 3],
        [4, 5, 6]]
```

`m[0]` is the first row of the table, `m[1]` is the second row, and so on. `m[r]` refers to the row with the index `r` (the (`r`+1)-th row of the table — recall that in Python indices start from 0). The elements in the row with the index `r` are `m[r][0]`, `m[r][1]`, `m[r][2]`, and so on. In the above example, the value of the element `m[0][2]` is 3.

1. Write a Python function that returns the sum of all the elements of a given matrix. Call the function *matrixSum*, it should take one argument, the matrix m and return a single value.

2. Write and test a Python function that returns the sum of the elements on the main diagonal (upper left to lower right) of a square matrix (represented as a list of lists). (In linear algebra, this value is called the *trace* of the matrix.)

3. An *n* by *n* matrix defines a *linear transformation* (function) on *n*-dimensional vectors. If $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$ and $\vec{x} = \left(x_1, x_2, ..., x_n\right)$, then $A \cdot \vec{x}$ is a new vector $\vec{y} = \left(y_1, y_2, ..., y_n\right)$, such that $y_i = a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n$. In other words, y_i is the dot product of the *i*-th row of the matrix and $\vec{x}$. Write and test a Python function that takes an *n* by *n* matrix *A* and an *n*-dimensional vector $\vec{x}$ and returns the vector $A \cdot \vec{x}$.

4. Suppose we have two *n* by *n* matrices:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \text{ and } B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & & & \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix}$$

The matrix *C* in which $c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + ... + a_{in}b_{nj}$ is called the *product* of *A* and *B* and denoted as $A \cdot B$ or simply AB. c_{ij} is the dot product of the *i*-th row in *A* and the *j*-th column in *B*. Write and test a Python function that takes two square matrices of the same size and returns their product. Hint: don't forget to create the resulting matrix before you put values into it.