CSE 20
Beginning Programming in Python
Programming Assignment 8

The goal of this project is to write a Python module called Vector that provides functions for performing some standard vector operations, detailed below. Vectors in this project will be represented by tuples of numbers, either floats or ints. Your module will be stored in a file called `Vector.py`. Begin by copying the files `VectorStub.py` and `VectorTest.py` from the examples section of the class webpage. Rename the file `VectorStub.py` to `Vector.py` and begin filling in the 11 missing function definitions.

Five of these functions (`add()`, `sub()`, `hadamard()`, `dot()` and `angle()`) will take two vectors u and v as input. If u and v are not of the same dimension, the function will raise a `ValueError()` containing the message: `'incompatible vectors: '+str(u)+', '+str(v)`. Follow the example Matrix.py on the class webpage to see how this might be done.

The word "length" could have two different meanings in this context. On the one hand we speak of the *length of a tuple* to mean its number of elements. On the other hand the *length of a vector* is commonly understood to be the geometric length of the directed line segment it represents. The number of components in a vector is more properly called its *dimension*. In the following descriptions of required vector operations we use *length* and *dimension* in this geometric sense.

`add(u, v)`
Returns the elementwise sum of the two vectors.
Example: `add( (1, 3, -5), (2, -2, 1) ) = (3, 1, -4)`

`negate(u)`
Returns the elementwise negation of a vector.
Example: `negate( (2, -2, 1) ) = (-2, 2, -1)`

`sub(u, v)`
Returns the elementwise difference of two vectors.
Example: `sub( (1, 3, -5), (2, -2, 1) ) = (-1, 5, -6)`

`scalarMult(c, u)`
Returns the elementwise product of a vector by the number c.
Example: `scalarMult( 2, (1, 3, -5) ) = (2, 6, -10)`

`hadamard(u, v)`
Returns the elementwise product of two vectors, also called the *Hadamard Product*.
Example: `hadamard( (1, 3, -5), (2, -2, 1) ) = (2, -6, -5)`

`dot(u, v)`
Returns the sum of the elements of the Hadamard product of two vectors, called the *Dot Product*.
Example: `dot( (1, 3, -5), (2, -2, 1) ) = -9`

`length(u)`
Returns the (geometric) length of a vector, i.e. the square root of `dot(u, u)`.
Example: `length( (2, -2, 1) )` $= \sqrt{4 + 4 + 1} = \sqrt{9} = 3.0$

`dim(u)`
Returns the dimension of a vector, i.e. its number of elements.
Example: `dim( (1, 3, 5) ) = 3`

`unit(v)`
Returns a *unit vector* (one whose geometric length is 1) in the direction of u. To compute this quantity, scalar multiply the vector by the reciprocal of its length.
Example: `unit( (2, -2, 1) ) = (0.6666.., -0.6666.., 0.3333..)`

`angle(u, v)`
Returns the *angle* between two vectors. To compute this function use the formula $\cos^{-1}(\hat{u} \cdot \hat{v})$, where $\cos^{-1}()$ is the inverse cosine function (denoted by `acos()` in the math module), $\hat{u}$ and $\hat{v}$ are unit vectors in the direction of u and v respectively, and $\hat{u} \cdot \hat{v}$ is their dot product.
Example: `angle( (1, 3, -5), (2, -2, 1) ) = 2.1026..`

`randVector(n, a, b)`
Returns a vector of dimension `n` whose elements are random floats in the range $[a, b)$. Use the function `uniform()` in the random module to generate the random components returned by this function.

Each of the above functions should include a doc string describing its operation. This doc string should be composed in such a way that a call to `help(Vector)` in Python interactive mode, produces the following output.

```
>>> import Vector
>>> help(Vector)
Help on module Vector:

NAME
    Vector

DESCRIPTION
    This module provides functions to perform standard vector operations.  Vectors
    are represented as tuples of numbers (floats or ints).  Functions that take two
    vector arguments will raise a ValueError() exception if the two vectors are of
    different dimensions.

FUNCTIONS
    add(u, v)
        Return the vector sum u+v.

    angle(u, v)
        Return the angle (in radians) between vectors u and v.

    dim(u)
        Return the dimension of the vector u.

    dot(u, v)
        Return the dot product of u with v.

    hadamard(u, v)
        Return the Hadamard product of u with v.

    length(u)
        Return the (geometric) length of the vector u.
```

```
    negate(u)
        Return the negative of the vector u.

    randVector(n, a, b)
        Return a vector of dimension n whose components are random floats in
        the range [a, b).

    scalarMult(c, u)
        Return the scalar product cu of the number c by the vector u.

    sub(u, v)
        Return the vector difference u-v.

    unit(v)
        Return a unit (geometric length 1) vector in the direction of v.

FILE
    c:\users\ptantalo\documents\code\cse20\fall20\solutions\pa8\vector.py
```

The only exception to this output will be the path quoted in the `FILE` section at the end. Otherwise the output from your module will match the above exactly.

Test your module by placing the files Vector.py and VectorTest.py in the same directory, then running VectorTest as a script. Other than the random vector at the end, its output should be identical to that given below.

```
$ python3 VectorTest.py

dim( (-3, -4, 7) ) = 3
dim( (4, 4) ) = 2
(-3, -4, 7) + (6, -2, 2) = (3, -6, 9)
- (6, -2, 2) = (-6, 2, -2)
(-3, -4, 7) - (6, -2, 2) = (-9, -2, 5)
2.5 (-3, -4, 7) = (-7.5, -10.0, 17.5)
-3.5 (6, -2, 2) = (-21.0, 7.0, -7.0)
hadamard( (-3, -4, 7) , (6, -2, 2) ) = (-18, 8, 14)
dot( (-3, -4, 7) , (6, -2, 2) ) = 4
| (-3, -4, 7) | = 8.602325267042627
| (6, -2, 2) | = 6.6332495807108
unit( (4, 4) ) = (0.7071067811865475, 0.7071067811865475)
unit( (-2, 2) ) = (-0.7071067811865475, 0.7071067811865475)
angle( (4, 4) , (-2, 2) ) = 1.5707963267948966

random vector =  (-0.5101713311427276, 0.04282108893167491)

$
```

What to turn in

Submit the file `Vector.py` to the assignment name `pa8` on Gradescope. As always start early and ask questions if anything is not clear.