CMPS 5P
Introduction to Programming in Python
Programming Assignment 7

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 lists 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 nine missing function definitions. Note that one definition, `angle()` which you should not change, is already included. Thus the module will contain ten vector operations in all.

Some of the required functions (`add()`, `sub()`, `zip()`, `dot()` and `angle()`) will take two vector arguments as input. These functions are not well defined when their two arguments are incompatible (i.e. have different list lengths). All functions will return either a new vector (a list object) or a number (a float or an int). None of the functions will alter their list arguments. The word "length" could have two different meanings in this project. On the one hand we speak of the *length of a list* meaning its number of elements. On the other hand the *length of a vector* is commonly understood to mean 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 will use *length* and *dimension* in this geometric sense.

`add(u, v)`
This function will return the sum `u+v` of the two vectors `u` and `v`. To do this just add corresponding elements in the two lists.
Example: `[1, 3, -5] + [2, -2, 1] = [3, 1, -4]`

`negate(u)`
Returns the negative `−u` of the vector *u*.
Example: `-[2, -2, 1] = [-2, 2, -1]`

`sub(u, v)`
Returns the difference u-v of the two vectors. You may use add() in conjunction with negate() to accomplish this.
Example: `[1, 3, -5] - [2, -2, 1] = [-1, 5, -6]`

`scalarMult(c, u)`
Returns the result `cu` of multiplying the number `c` by the vector `u`. Each element of `u` is multiplied by `c`.
Example: `2[1, 3, -5] = [2, 6, -10]`

`zip(u, v)`
This is not a standard vector operation, but is useful in computing `dot()` below. Function `zip()` returns the component-wise product vector of `u` with `v`, i.e. multiply corresponding elements in the two lists.
Example: `zip( [1, 3, -5] , [2, -2, 1] ) = [2, -6, -5]`

`dot(u, v)`
Returns the sum of the elements in `zip(u, v)`.
Example: `dot( [1, 3, -5] , [2, -2, 1] ) = -9`

`length(u)`

Returns the (geometric) length of the directed line segment represented by vector `u`. To compute this quantity take the square root of the sum of the squares of the components of `u`, or more succinctly the square root of `dot(u, u)`.

Example: $\text{length}\,([1, 3, -5]) = \sqrt{1^2 + 3^2 + (-5)^2} = \sqrt{35} \approx 5.916$

`unit(v)`

Returns a unit vector (i.e. a vector whose geometric length is 1) having the same direction as `u`. To compute this quantity, scalar multiply `u` by the reciprocal of its length.

Example: $\text{unit}\,([1, 3, -5]) = [\frac{1}{\sqrt{35}}, \frac{3}{\sqrt{35}}, \frac{-5}{\sqrt{35}}] \approx [0.169, 0.507, -0.845]$

`angle(u, v)`

This function returns the angle between vectors `u` and *v*, and is already defined in `VectorStub.py`. Do not change it.

`randVector(n, a, b)`

Returns a vector of dimension `n` (i.e. list length `n`) whose components are random floats in the range $[a, b)$. Note that the random module contains a method called random() (small r) returning a random float in the range $[0, 1)$. Use random.random() 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)` 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 lists of numbers (floats or ints).  Functions that take two
    vector arguments may give arbitrary output if the vectors are not compatible,
    i.e. of the same dimension.

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

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

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

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

    negate(u)
        Return the negative -u 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.

    zip(u, v)
        Return the component-wise product of u with v.

FILE
    c:\users\ptantalo\desktop\programs\python3.3\cmps 5p\programs\pa6\vector.py
```

The only exception to this output should 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. The command `python VectorTest.py` should produce the following output.

```
[-3, -4, 7]
[6, -2, 2]
[3, -6, 9]
[-6, 2, -2]
[-9, -2, 5]
[-7.5, -10.0, 17.5]
[-21.0, 7.0, -7.0]
[-18, 8, 14]
4
8.602325267042627
6.6332495807108
[-0.34874291623145787, -0.46499055497527714, 0.813733471206735]
[0.9045340337332909, -0.30151134457776363, 0.30151134457776363]
85.98027117814442
[0.0, 0.0, 0.0]
```

What to turn in

Submit the file `Vector.py` to the assignment name `pa7` in the usual way. As always start early and ask questions if anything is not clear.