You are only required to submit your solution for one of the following: 3W (this assignment) or 3F

PIC 16, Winter 2018 – Assignment 3W

Assigned 1/24/2018. Code (a single `.py` file) due by the end of class 1/29/2018 on CCLE. Hand in a printout of this document with the self-assessment portion filled out by the end of class on 1/29/2018.

In this homework, you will create a class named `MathVector` to represent an *N*-dimensional mathematical (Euclidean) vectors. If you want to do this assignment but aren't familiar with mathematical vectors, let me know if you find Vectors – The "How" and Vectors – The "Why" to be helpful.

Tasks

1. Write an initializer.
    - If there is a single numerical argument (assumed to be an integer), the `MathVector` should represent a zero vector (vector of all zero components) of the specified number of dimensions
    - If there is a single list or tuple argument or if there are multiple arguments (assumed to be numerical), the `MathVector` should represent a vector with the specified (Cartesian) components.
    - The constructor does not need to work properly for any other input cases. If the user thinks that a string or a dictionary can be converted into a euclidean vector meaningfully, that's their problem.
2. Write methods. None of these should modify the operands.
    - `get_el` – returns the *i*-th component of the vector (1-indexed)
    - `neg` – returns the negative of the original vector (leaving the original unchanged)
    - `mag` – returns the magnitude of the vector
    - `dot` – returns the dot product of the vector and another vector
    - `plus` – returns the sum of the vector and another vector
    - `sp` – returns the product of the vector and a scalar ("scalar product")
    - `print_me` – prints a representation of the vector to the console like "`[1, 0, 0]`"
3. Write magic methods so that built in operators and functions [ ], - (negation), `abs` (should take the magnitude), *, and + call the appropriate methods above. Also, make `print x` work if `x` is a `MathVector`; try doing this by overloading `str` rather than `print`. Note that the * operator should not only work for two `MathVector`s but also for a scalar *before or after* the `MathVector`.

Hint: most of your functions should be only one line long! In my solution, only `get_el`, `__mul__`, and `__init__` are longer! You can use list comprehension to simplify functions like `neg`, `plus`, `mag`, and `dot`. (It would be better to use *generator expressions* for those last two. Generator expressions are faster and require less memory because they don't generate a whole list and store it in memory, rather they generate each element as it is needed in the computation… We'll see these soon!)

For the methods, please use the exact names above and make sure your class makes the provided test code in `Assignment 2F Tester.py` work (when the test code is added to your file or you import your class).

You can assume that the user is nice with inputs. For instance, the inputs to `dot` and `plus` methods are `MathVectors` of the same dimension.

Self-Assessment
10 points for each of the first three lines of test-code output that matches `Assignment 6 Example.txt` and 5 points for each additional matching line.

Score: