PHY1112: Assignment 4

> Be classy and roll with it

Assigned: January 30th, 2024

Due: February 6th, 2024

Learning Objectives

1. Making new methods for your classes
2. Practice importing functions from other modules

Grade Breakdown

Part	1	2	Total
Points	15	16	31
Score			

Question 1: Stay classy.

a) Write a new custom class, `Vector3`, which is like the `Vector2` custom class you wrote in lab, but now is three dimensional with `x`, `y`, and `z` components (i.e., data attributes).

Write `__str__()` and `magnitude()` methods for your `Vector3` class, similar to what you did for your `Vector2` class in the lab. The formula for the magnitude in three dimensions is $\sqrt{x^2 + y^2 + z^2}$.

Demonstrate they work with the following vector, copying a screenshot of your results into your assignment solutions document:

$$\vec{A} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

(4 marks)

b) Write two special methods `__add__()` and `__sub__()` that perform, respectively, vector addition and subtraction between the first `Vector3` instance and a second `Vector3` instance. They are invoked by Python when the + and – operators are used with `Vector3` objects.

Demonstrate that your `__add__()` and `__sub__()` methods work with the following vectors, copying a screenshot of your results into your assignment solutions document:

$$\vec{A} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad \vec{B} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

(3 marks)

c) Write another method called `inner_product()`, that performs an inner product between the first `Vector3` instance and a second (sometimes called the dot product).

Demonstrate that it works using the $\vec{A}$ and $\vec{B}$ vectors above, copying a screenshot of your results into your assignment solutions document.

(2 marks)

d) Using your `inner_product()` and `__add__()` methods, show that the following is true:

$$\left(\vec{A} + \vec{B}\right) \cdot C = \vec{A} \cdot \vec{C} + \vec{B} \cdot \vec{C}.$$

Copy a screenshot of your results into your assignment solutions document. Use the $\vec{A}$ and $\vec{B}$ vectors as above, as well as

$$\vec{C} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

(2 marks)

e) Write another method that calculates the angle between two `Vector3` objects and call it `angle_between()`. Recall that the formula for the angle between two vectors $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ is:

$$\cos^{-1} \frac{\overrightarrow{v_1} \cdot \overrightarrow{v_2}}{|\overrightarrow{v_1}||\overrightarrow{v_2}|},$$

where the dot operator indicates the inner product, and the | | indicates you are taking the magnitude. Hint: This will also require the math module.

What is the angle between $\vec{A}$ and $\vec{B}$? Give the answer in both in radians and degrees, and copy a screenshot of your results in your assignment solutions document.

(3 marks)

(15 marks total, 1 for doc strings/file header/variable naming/comments)

Question 2: Just roll with it.

In this question, we will make use of the `time` and `itertools` modules to compute how long it takes to do task via nested loops versus via vectorization.

Consider four N-sided dice being rolled, where N is something that will vary so should be implemented as a variable you can change the value of each time you run your program.

a) Using four nested for loops, create a list of every possible combination of dice rolls. The entries of this list should be tuples, where the tuples each contain four numbers (that is, one number for each die that is rolled).

   Report how many combinations there are for four 20-sided dice (that is N=20).

   (3 marks)

b) Inside your four for loops from part a, also create a list that contains as entries the sum value of the four dice for each combination. That is, add up the four numbers in each of your tuples and put all the sums in a new list.

   (2 marks)

c) Write code to determine the average of all the sums, and run it for 20-sided dice. Report your answer including a screenshot.

   (1 mark)

d) Now, do the same thing, but with vectorization. First, use `itertools.product()` to generate your list of all possible combination of dice rolls, where the entries in the list are tuples each containing the four numbers.

   Report how many combinations there are for four 20-sided dice (that is N=20)

   (3 marks)

e) As in part b, using this list of tuples, create a list that contains as entries the sums of all the combinations, that is, the sum value of numbers in each tuple. Accomplish this using at most one for loop.[1]

   (2 marks)

[1] Note that you can use list comprehension here to avoid an official loop altogether, but we will not be covering that officially in this course.

f) Determine the average of all the sums with no loop. Again, use N=20 and put a screenshot of your results into your assignment solution document.

(1 mark)

g) Use the `time` module to determine how long it took to run the calculations in the nested for-loop method (parts a,b,c) and how long it took to run the calculations with vectorization (parts d,e,f).

Run your code three times for N=10. You should now have three time measurements for the for nested loop method, and three time measurements for the vectorization method.

Use these to calculate an average run time for nested for loops, as well as an average run time for vectorization, and record them in the table below.

Repeat for N=20 and N=50, continuing to fill in your results in the table below.

For each N, also report the number of combinations.

Number of faces	Number of rolls	Number of combinations	Nested for loops average runtime	Vectorization average runtime
10	4			
20	4			
50	4			

Which method is faster? Comment on what you observed.

(3 marks)

(16 marks total, 1 for doc strings/file header/variable naming/comments)