Project 1: Framework 3

Submission Due: September 23 by 11:59PM

Program Description:

One of the most visually spectacular plays in basketball is the alley-oop dunk, where one player throws the ball into the air near the basket, and their teammate (or, occasionally, themselves) jumps, takes the ball in mid-air, and dunks it into the basket. Getting this play to work requires precise timing and aim for both the passer and the dunker, and while it is usually not a conscious calculation, every player that does it is using physics to figure out when and how to act. In this mini-project, you'll be more deliberately using physics to calculate how to keep an alley-oop from becoming an alley-oops. For simplicity, we will be confining ourselves to two dimensions, with the $+x$ direction being towards the basket, and the $+y$ direction being up.

For this project, you will be implementing a program that performs several calculations regarding the setup for an alley-oop dunk. You will need to prompt the user for the distance of the player with the ball from the hoop, in meters, the angle at which the ball is thrown, and the jumping player's jumping speed, in meters per second. Then, assuming the jumping player is 0.375 meters from the hoop, the jumping player's maximum reach is 2.35 meters, and that the ball is being thrown from a height of 1.5 meters, calculate the speed, in meters per second, at which the ball needs to be thrown and how long the jumping player must wait, in seconds, to catch the ball at the top of their jump for the alley-oop dunk.

Program Requirements

- As with all projects in this course, your program should have a comment at the top that includes your name, the date the program was created, and a brief description of the project, in your own words
- Import the `scipy.constants` library
- Import the `numpy` library
- Define a new constant variable and initialize it to the value of `scipy`'s gravity constant
    - You can use `scipy.constants.g` to access the value of gravity
- Define a new variable to store the jumping player's distance from hoop, and initialize it to the appropriate value
- Define a new variable to store the jumping player's maximum reach, and initialize it to the appropriate value
- Define a new variable to store the height from which the ball is being thrown, and initialize it to the appropriate value
- Prompt the user for the following values and store each in new variables you define:
    - The distance of the player with the ball from the hoop, in meters
    - The angle at which the ball is thrown
        - Do not forget to convert from degrees to radians! You can use `numpy`'s `numpy.pi` to access the value of pi
    - The jumping player's jumping speed, in meters per second

- Using the formula you derive from the framework, calculate the velocity of the ball, in meters per second, and store it in a new variable you define
    - `numpy` provides access to the `sqrt`, `cos`, `sin`, and `tan` functions
- Using the formula you derive from the framework, calculate the how long the jumping player must wait, in seconds, and store it in a new variable you define
- Using an appropriate message, output the velocity of the ball, in meters per second, using two decimal points of precision
- Using an appropriate message, output how long the jumping player must wait, in seconds, using two decimal points of precision
- You are welcome to define any additional variables to store intermediate steps of the equations as you see necessary
- Your code should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, date, and brief description), comments for each variable, and commented blocks of code
- Your program source code should be named **alleyoop.py**

Submission

- Your program will be graded largely upon whether it works correctly
- Your program will also be graded based upon your program style. This means that you should use comments (as directed) and meaningful variable names
- You must submit the **alleyoop.py** file to Canvas by the due date and time
- Projects are meant to be individual coding assignments, so no collaboration is allowed. This includes downloading code off of the internet. Any discovered instances of this will be considered cheating and appropriate actions will the taken according to the course syllabus
- Be sure that you have tested the version of the program you wish to submit to make sure it works correctly. You will not be allowed to resubmit work after the deadline that does not have a third-party timestamp such as Dropbox or an email. Timestamps generated by your operating system will not be acceptable.

Sample Output:

```
How far from the hoop is the player with the ball, in meters: 6
What is jumping player's jumping speed: 5
What angle is the ball thrown at: 55
The speed the ball must thrown is 8.93 m/s.
The jumping player needs to wait 0.59 seconds after you throw the ball to
catch it at the perfect moment.

How far from the hoop is the player with the ball, in meters: 4.5
What is jumping player's jumping speed: 4.75
What angle is the ball thrown at: 60
The speed the ball must thrown is 8.05 m/s.
The jumping player needs to wait 0.54 seconds after you throw the ball to
catch it at the perfect moment.
```

Bonus:

For 5 bonus points, you will be creating a visualization of the scenario in this project and framework! Note, you should not begin the bonus until you have fully implemented the required functionality for the project.

First, you will need to copy the code in the `bonus.py` file into your program after the end of your code. Then, you will need to calculate the amount of time that elapses between when the ball is thrown and when it is caught by the jumping player, and store that in a new variable you define. Then for each of the 10 variables (`d, l, hb, hs, thb, vb, vs, hh, tcatch, tj`) you will need to assign a value from the appropriate variable you created. For example, `thb` should be assigned a value from the variable you are using to store the angle the ball is thrown in radians. This idea of adding someone else's functionality to yours and then connecting your variables to their inputs is actually quite common and we will see it a bit more when we work with functions and later on you will see it in more advanced situations using application programming interfaces (APIs). Finally, you will need to add the following code to the top of your program just below where you import `numpy` and `scipy.constants`:

```
from vpython import box, vector, rate, color, sphere, ring
```

This code is required for the visualizations to properly be executed.