Week 8 Assignment

01219114/115 Programming I

Submission

1. Create *StudentID_Firstname*_Wk8 folder, where *StudentID* is your KU student ID and *Firstname* is your given name.
2. Place files to submit, `vector.py`, `ball.py`, and `app_ball.py`, in the folder.
3. Compress the folder and name your compressed file *StudentID_Firstname*_Wk8.zip, then submit it in Google Classroom by the submission deadline.
4. Submit a summary text file (filename: `summary.txt`). In this summary, tell us what you have completed and what you have not. Submit in Google Classroom as well.

Grading Criteria

1. Correctness (75%): Your program must pass all the doctests and give expected results.
2. Cleanliness (25%): Your program must follow the PEP8 convention. Variable names are meaningful. Docstrings are written for all methods and functions. Comments are added at certain points for others to understand your code easily.

Your Task

You are to implement an application that simulates a ball's motion in a two-dimensional space. The ball's movement will follow a simple set of motion equations that rely on vector arithmetic.

The task can be broken down into three parts: the `vector` module, the `ball` module, and the application.

The **`vector`** module (`vector.py`)

The `vector` module is stored in the file `vector.py`. It contains the `Vector` class, which has already been written as part of the in-class Elab exercises. Please make sure that the code passes all the doctests provided in the template code.

The **`ball`** module (`ball.py`)

The ball module is stored in the file `ball.py`. It contains the `Ball` class that will be used to construct Ball objects. Ball objects store their attributes as Vector objects, so the `Vector` class needs to be imported from the `vector` module using the following statement:

```
from vector import Vector
```

Each Ball object contains three attributes:

1. `pos` - a Vector object representing the current position of the ball
2. `vel` - a Vector object representing the current velocity of the ball
3. `acc` - a Vector object representing the acceleration of the ball

Each Ball object has three methods:

1. **`__init__(self, pos, vel, acc)`** - initializes a Ball object with the initial vectors for the `pos`, `vel`, and `acc` attributes.
2. **`update(self, dt)`** - updates the `pos` and `vel` attributes to the values after a duration of `dt` seconds has passed. You can calculate the new values for `pos` and `vel` using the following two equations of motion:

$$\mathbf{vel}_{new} = \mathbf{vel} + \mathbf{acc} \cdot dt$$
$$\mathbf{pos}_{new} = \mathbf{pos} + \mathbf{vel}_{new} \cdot dt$$

3. **`__repr__(self)`** - returns a string representation of the object in the following form:

```
Ball(pos=<vector>, vel=<vector>, acc=<vector>)
```

where each *<vector>* is the string representation provided by the `Vector` class

After you are done with the implementation, make sure the code passes all the doctests provided in the template code.

The application (`app_ball.py`)

The final part of our task is to implement a simple console-based program, `app_ball.py`, that utilizes the Ball and Vector classes. (The next version will be implemented as a graphical program.) It will simulate movement of a single Ball object by asking the user to provide the following inputs:

- The time step (in seconds) to perform the simulation
- The time limit (in seconds) the simulation will run
- The initial position vector of the ball
- The initial velocity vector of the ball
- The initial acceleration vector of the ball

The program then starts the simulation and reports the position and velocity of the ball every time step with two decimal places until the time limit is reached.

Since the application makes use of the Ball class and the Vector class, you will have to import both classes from their respective modules using these statements.

```
from vector import Vector
from ball import Ball
```

Sample Output #1

```
Enter the time step in seconds: 0.2
Enter the time limit in seconds: 3
Enter the ball's initial position vector (x,y): 0,0
Enter the ball's initial velocity vector (x,y): 1,2
Enter the ball's acceleration vector (x,y): 0,0
Time=0.00 sec, pos=(0.00,0.00), vel=(1.00,2.00)
Time=0.20 sec, pos=(0.20,0.40), vel=(1.00,2.00)
Time=0.40 sec, pos=(0.40,0.80), vel=(1.00,2.00)
Time=0.60 sec, pos=(0.60,1.20), vel=(1.00,2.00)
Time=0.80 sec, pos=(0.80,1.60), vel=(1.00,2.00)
Time=1.00 sec, pos=(1.00,2.00), vel=(1.00,2.00)
Time=1.20 sec, pos=(1.20,2.40), vel=(1.00,2.00)
Time=1.40 sec, pos=(1.40,2.80), vel=(1.00,2.00)
Time=1.60 sec, pos=(1.60,3.20), vel=(1.00,2.00)
Time=1.80 sec, pos=(1.80,3.60), vel=(1.00,2.00)
Time=2.00 sec, pos=(2.00,4.00), vel=(1.00,2.00)
Time=2.20 sec, pos=(2.20,4.40), vel=(1.00,2.00)
Time=2.40 sec, pos=(2.40,4.80), vel=(1.00,2.00)
Time=2.60 sec, pos=(2.60,5.20), vel=(1.00,2.00)
Time=2.80 sec, pos=(2.80,5.60), vel=(1.00,2.00)
Time=3.00 sec, pos=(3.00,6.00), vel=(1.00,2.00)
```

Sample Output #2

```
Enter the time step in seconds: 0.1
Enter the time limit in seconds: 2
Enter the ball's initial position vector (x,y): 0,10
Enter the ball's initial velocity vector (x,y): 5,0
Enter the ball's acceleration vector (x,y): 0,-9.8
Time=0.00 sec, pos=(0.00,10.00), vel=(5.00,0.00)
Time=0.10 sec, pos=(0.50,9.90), vel=(5.00,-0.98)
Time=0.20 sec, pos=(1.00,9.71), vel=(5.00,-1.96)
Time=0.30 sec, pos=(1.50,9.41), vel=(5.00,-2.94)
Time=0.40 sec, pos=(2.00,9.02), vel=(5.00,-3.92)
Time=0.50 sec, pos=(2.50,8.53), vel=(5.00,-4.90)
Time=0.60 sec, pos=(3.00,7.94), vel=(5.00,-5.88)
Time=0.70 sec, pos=(3.50,7.26), vel=(5.00,-6.86)
Time=0.80 sec, pos=(4.00,6.47), vel=(5.00,-7.84)
Time=0.90 sec, pos=(4.50,5.59), vel=(5.00,-8.82)
Time=1.00 sec, pos=(5.00,4.61), vel=(5.00,-9.80)
Time=1.10 sec, pos=(5.50,3.53), vel=(5.00,-10.78)
```

```
Time=1.20 sec, pos=(6.00,2.36), vel=(5.00,-11.76)
Time=1.30 sec, pos=(6.50,1.08), vel=(5.00,-12.74)
Time=1.40 sec, pos=(7.00,-0.29), vel=(5.00,-13.72)
Time=1.50 sec, pos=(7.50,-1.76), vel=(5.00,-14.70)
Time=1.60 sec, pos=(8.00,-3.33), vel=(5.00,-15.68)
Time=1.70 sec, pos=(8.50,-4.99), vel=(5.00,-16.66)
Time=1.80 sec, pos=(9.00,-6.76), vel=(5.00,-17.64)
Time=1.90 sec, pos=(9.50,-8.62), vel=(5.00,-18.62)
Time=2.00 sec, pos=(10.00,-10.58), vel=(5.00,-19.60)
```

Hints

- The `str.split()` method can be used to split a coordinate pair into two numbers.
- Your program may experience a floating-point error problem, especially during the time limit check. The `round()` function can be used to mitigate this problem.