Assignment 03

***Due Date**: As per the deadline on Avenue dropbox titled 'A03'. Time required to do this assignment is about 1 hour. The deadline is set to the end of the week to give students with special accommodations adequate time to complete the assignment. No further extensions will be given.*

Part A: Dropping a Ball!

We all drop the ball occasionally (pun intended)☺ Let us study how it bounces back though.

The process: Assume that the ball is dropped from a certain height (h) with an initial speed of u m/s. The ball travels down and as it is acted upon by the gravitational force (g ~ 10m/s^2), it hits the ground with a higher final speed v. This final speed is calculated using the relation v = u+gt. Upon hitting the ground the ball rebounds. Let us assume that it rebounds without any loss of energy, i.e., it rebounds with the same speed v. As the ball moves up, it decelerates at the rate of -g (i.e. g ~ -10m/s^2) and eventually when it get back to the original height of h, it stops. It then starts falling back to the ground, emulating the initial fall. The distance that the ball rises/falls is d = ut + 0.5gt^2. (Note: If the initial dropping velocity is 0 m/s then the ball must rise to the exact same initial height. If the ball it thrown with a certain downward speed, then it rebounds to a greater height than the initial height.)

Now, write a python program that simulates this process by doing the following: Using the **getInput**() function obtain the following values from the user: initial height and initial speed. (Both will be a positive number). Use the **motionSimulator**() function to simulate the falling and rising of the ball. In this you will have to increment time by small steps of 0.04s and determine the precise position of the ball for these timesteps, plotting it as a function of time. You must print the following data onto the screen each time the ball reaches the extremum:

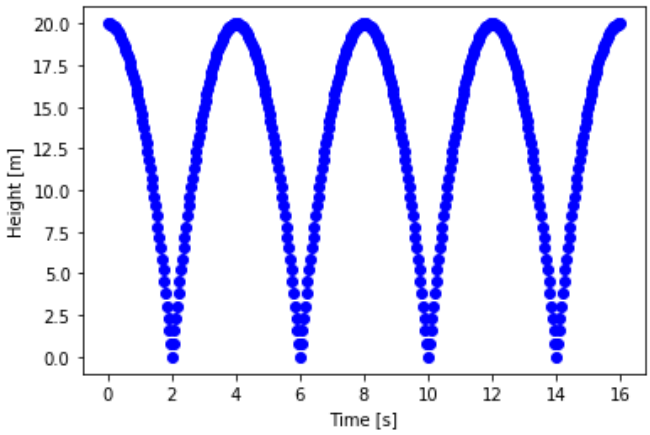
1. Position (Top or Bottom indicated by T and B, respectively).
2. The speed at that position
3. Total number of timesteps to be simulated (use 401 timesteps)
4. The time interval needed to make that motion from the previous extremum.
5. The total time that the ball has been bouncing for since the beginning of the simulation.
6. The cumulative number of 0.04s timesteps that have been simulated.
7. A graph showing the bouncing of the ball through the entire duration
   Hint: You will need the pyplot library to plot graphs. While a detailed introduction to pyplot is in https://matplotlib.org/stable/tutorials/introductory/pyplot.html, a relevant code that you need for this exercise is:

```
import matplotlib.pyplot as plt
plt.plot(time,current_height,'bo')
plt.xlabel('Time [s]')
plt.ylabel('Height [m]')
```

Use floats for all your variables except the number of timesteps. **The output should be printed only from the main program. Your program output should be in the following format:**

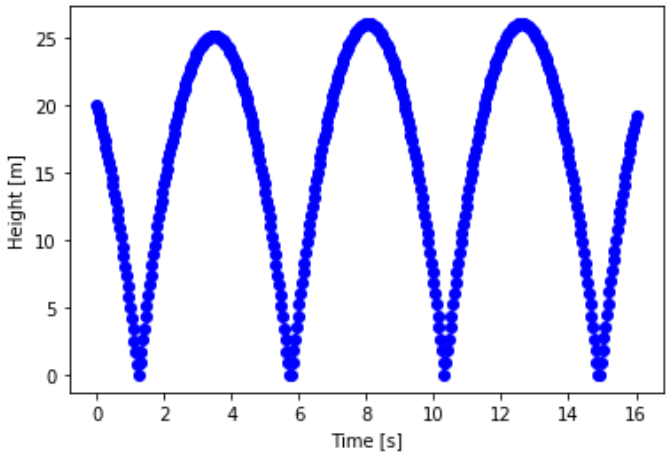
Test Case 1: (h=20, u=0)

```
B: 20.00  2.00  2.00    50
T:  0.00  2.00  4.00   100
B: 20.00  2.00  6.00   150
T:  0.00  2.00  8.00   200
B: 20.00  2.00 10.00   250
T:  0.00  2.00 12.00   300
B: 20.00  2.00 14.00   350
T:  0.00  2.00 16.00   400
```



Test Case 2: (h=20, u=10)

```
B: 22.40  1.24  1.24    31
T:  0.00  2.24  3.48    87
B: 22.80  2.28  5.76   144
T:  0.00  2.28  8.04   201
B: 22.80  2.28 10.32   258
T:  0.00  2.28 12.60   315
B: 23.20  2.32 14.92   373
```



Part B: Energy Loss

There is an unrealistic assumption in part A. The ball will always lose some energy when it hits the ground. As a result it will not be able to get back to the same height. So, let us fix the problem. Copy and paste the code from part A since you will need most of it. Use the **energyLoss**() function that determines the

rebound speed (v) using the principle of conservation of energy. For a given bounce, when the ball is at the highest point (y), its potential energy is E_potential = mgy, where m is the mass of the ball (not needed), and g is as in part A. When it hits the ground, this energy converts itself to the kinetic energy E_kinetic = $0.5mv^2$, where v is the rebound speed. Assume that the loss of the potential energy is Ep when the ball hits the ground and calculate the rebound speed (v) in this function. Ep is obtained from the **getInput()** function as a decimal value (example, Ep = 0.1 for 10% loss).

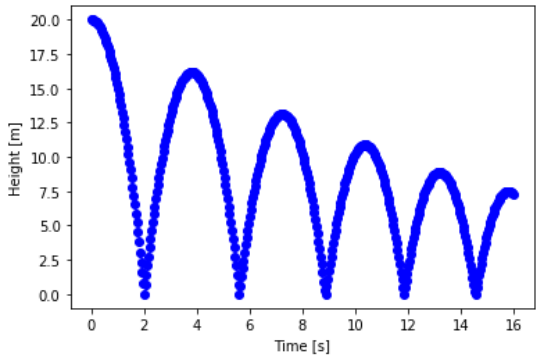
As in part A, using the **getInput**() function obtain the following values from the user: initial height and initial speed. (Both will be a positive number). Use the **motionSimulator**() function to simulate the falling and rising of the ball. In this you will have to increment time by small steps of 0.04s and determine the precise position of the ball for these timesteps, plotting it as a function of time. You must print the following data onto the screen each time the ball reaches the extremum:

1. Position (Top or Bottom indicated by T and B, respectively).
2. The speed at that position
3. Total number of timesteps to be simulated (use 401 timesteps)
4. The time interval needed to make that motion from the previous extremum.
5. The total time that the ball has been bouncing for since the beginning of the simulation.
6. The cumulative number of 0.04s timesteps that have been simulated.
7. A graph showing the bouncing of the ball through the entire duration.

The output should be printed only form the main program. Try different combinations of inputs to understand how the ball bounces. **Your program output should be in the following format:**

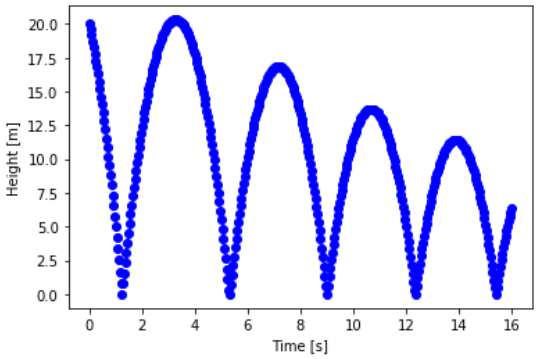
Test Case 1: (h=20, u=0, Ep = 0.1)

```
B: 18.00  2.00  2.00    50
T:  0.00  1.80  3.80    95
B: 16.20  1.80  5.60   140
T:  0.00  1.64  7.24   181
B: 14.76  1.64  8.88   222
T:  0.00  1.48 10.36   259
B: 13.32  1.48 11.84   296
T:  0.00  1.36 13.20   330
B: 12.24  1.36 14.56   364
T:  0.00  1.24 15.80   395
```



Test Case 2: (h=20, u=10, Ep=0.1)

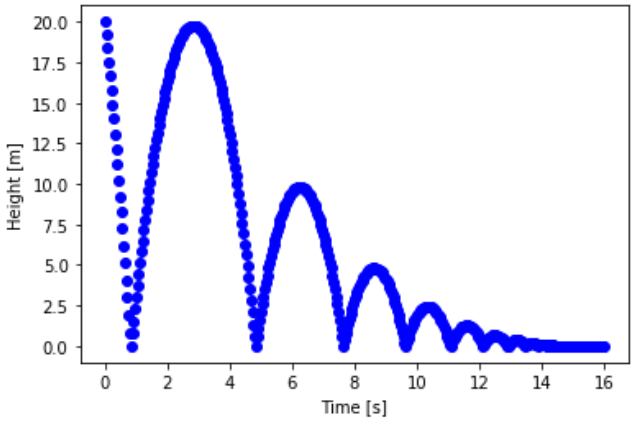
```
B: 20.16  1.24  1.24    31
T:  0.00  2.04  3.28    82
B: 18.36  2.04  5.32   133
T:  0.00  1.84  7.16   179
B: 16.56  1.84  9.00   225
T:  0.00  1.68 10.68   267
B: 15.12  1.68 12.36   309
T:  0.00  1.52 13.88   347
B: 13.68  1.52 15.40   385
```



Test case 3: (h=20,u=20,Ep=0.3)

```
B: 19.88  0.84  0.84    21
T:  0.00  2.00  2.84    71
B: 14.00  2.00  4.84   121
T:  0.00  1.40  6.24   156
B:  9.80  1.40  7.64   191
T:  0.00  1.00  8.64   216
B:  7.00  1.00  9.64   241
T:  0.00  0.72 10.36   259
B:  5.04  0.72 11.08   277
T:  0.00  0.52 11.60   290
B:  3.64  0.52 12.12   303
T:  0.00  0.40 12.52   313
B:  2.80  0.40 12.92   323
T:  0.00  0.28 13.20   330
B:  1.96  0.28 13.48   337
T:  0.00  0.20 13.68   342
B:  1.40  0.20 13.88   347
T:  0.00  0.16 14.04   351
B:  1.12  0.16 14.20   355
T:  0.00  0.12 14.32   358
B:  0.84  0.12 14.44   361
```

```
T:  0.00  0.12 14.56   364
B:  0.56  0.08 14.64   366
T:  0.00  0.08 14.72   368
B:  0.56  0.08 14.80   370
T:  0.00  0.08 14.88   372
B:  0.56  0.08 14.96   374
T:  0.00  0.08 15.04   376
B:  0.56  0.08 15.12   378
T:  0.00  0.08 15.20   380
B:  0.56  0.08 15.28   382
T:  0.00  0.08 15.36   384
B:  0.56  0.08 15.44   386
T:  0.00  0.08 15.52   388
B:  0.56  0.08 15.60   390
T:  0.00  0.08 15.68   392
B:  0.56  0.08 15.76   394
T:  0.00  0.08 15.84   396
B:  0.56  0.08 15.92   398
T:  0.00  0.08 16.00   400
```



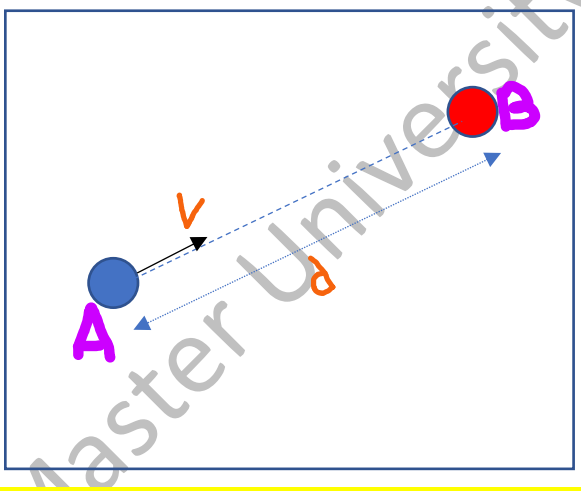
Submission Requirements: Submit a single word or PDF file containing the python program for each part along with the corresponding sample screenshots of the output when you run the program, to the dropbox titled **A03** in the *Assignments* section on Avenue. Do not submit a screenshot of the program itself. This will result in a grade of 0. Copy and paste your code into a text editor such as MS word instead. Only a screenshot of the program output (see sample of test case results including the figure) is allowed.

Assignment 2

***Due Date**: As per the deadline on Avenue dropbox titled 'A02'. Time required to do this assignment is about 1 hour. The deadline is set to the end of the week to give students with special accommodations adequate time to complete the assignment. No further extensions will be given.*

Suppose object A is traveling towards object B with a speed *v*. A and B are separated by a distance *d*. If A is decelerating by a value of *a* , we want to determine if it will hit the object B. The user must provide the following values *d, v* and *a*. *d* is in the range [5,10], *a* is in the range [-100,0] and v is in the range [1,10]. The distance travelled by the object A in time *t* (a positive number less than 10 that is also to be received from the user) can be calculated as

$$s = \max(0, vt + 0.5at^2)$$



If *s* is greater than or equal to *d* then the object will collide. Write a python program that does the following: (i) determines if the objects collide for a given set of *d, v*, *a* and t. (ii) for a given value of *d and v*, and starting with a = -50, determines the critical value of *a* at which A will just touch B.

Hint: To find the critical value of a, keep increasing *a* by a small amount (example, 0.2), and for each value of a scan through a long range of t to conclusively establish if the object A hits B or not. Note: All values are floating point data.

Submission Requirements: Submit a single word or PDF file containing the python program and sample screenshots of the output when you run the program to the dropbox titled **A02** in the *Assignments* section on Avenue. Do not submit a screenshot of the program itself. This will result in a grade of 0. Copy and paste your code into a text editor such as MS word instead. Only a screenshot of the program output is allowed.

Assignment 1

***Due Date**: As per the deadline on Avenue dropbox titled 'A01'. Time required to do this assignment is about 1 hour. The deadline is set to the end of the week to give students with special accommodations adequate time to complete the assignment. No further extensions will be given.*

1. Write a python program for the *Rock Paper Scissors Spock Lizard* game. The rules of the game are:

   Spock beats scissors and rock, but loses to paper and lizard.
   Lizard beats Spock and paper, but loses to rock and scissors.
   Rock beats scissors and lizard, but loses to paper and Spock.
   Paper beats rock and Spock, but loses to scissors and lizard.
   Scissors beats paper and lizard, but loses to rock and Spock.

   The program will have two players and each of them produce a random outcome, i.e., Rock, paper, scissor, spock, or lizard. Based on each player's outcome, determine the winner.

Submission Requirements: Submit a single word or PDF file containing the python program and sample screenshots of the output when you run the program to the dropbox titled **A01** in the *Assignments* section on Avenue. Do not submit a screenshot of the program itself. This will result in a grade of 0. Copy and paste your code into a text editor such as MS word instead. Only a screenshot of the program output is allowed.