SFWRTECH 3PR3:
Procedural and Objective Oriented Programming Concepts
(Assignment #3)

Student Name: Dojae Kim

Student Number: 400420323

Professor Name: Dr. Seshasai Srinivasan

Objective

The purpose of this Assignment 3:

1. To give students understanding python structure.
2. To write, test, and debug python programs.
3. To implement, and execute conditional statement iteratively until a given condition is satisfied.
4. To understand how to import various modules from python library.

Equation

Part A:

1. $V = u + gt$
2. $d = ut + 0.5gt^2$

Part B:

1. $E\_potential = mgy$
2. $E\_kinetic = 0.5mv^2$

Part A Source Code

```
# Student name: Dojae Kim
# Student number: 400420323
# Student email: kim408@mcmaster.ca
# Lecture: SFWRTECH 3PR3
# Assignment 3 Part A

from typing import Tuple
import matplotlib.pyplot as plt


def getInput() -> Tuple[float, float]:
    """
    Get input from user - tuple of 2 floats, 1st one is initial height and
    2nd one is initial speed
    :return: Tuple of initial values
    """
    done = False
    init_height = 0
    init_speed = 0
    while not done:
        try:
            init_height = float(input("Please enter initial height -> "))
            if init_height >= 0:
                done = True
```

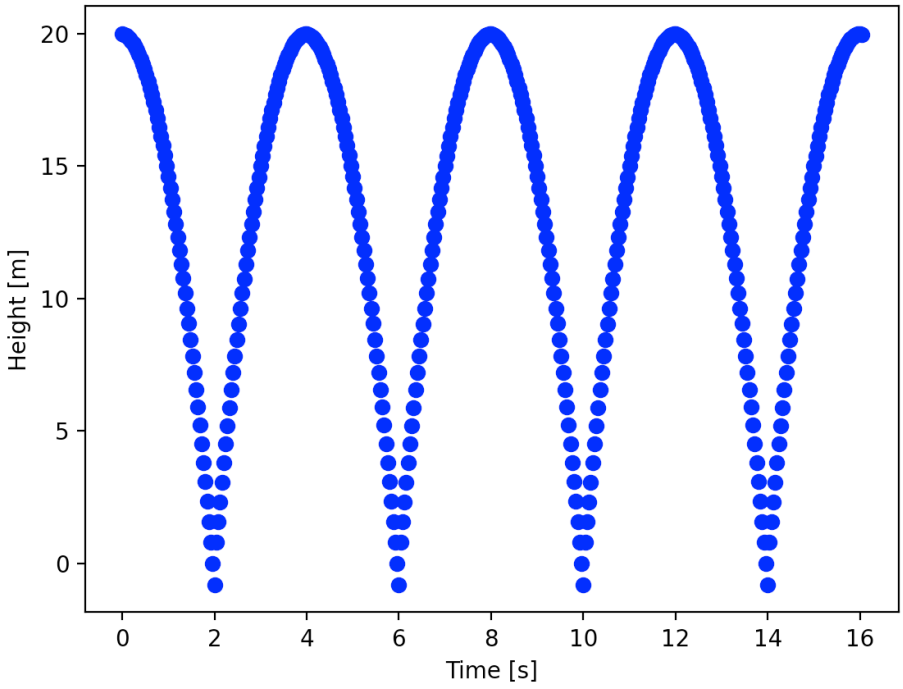
```
        except ValueError:
            print("Initial height should be positive!")

    done = False
    while not done:
        try:
            init_speed = float(input("Please enter initial speed -> "))
            if init_speed >= 0:
                done = True
        except ValueError:
            print("Initial speed should be positive!")

    return init_height, init_speed


def motionSimulator(init_values: Tuple[float, float]) -> None:
    """
    Simulate motion of the ball
    :param init_values: Initial values (height and speed)
    :return: None
    """
    down = True  # Initially ball travels down
    v = init_values[1]
    h = init_values[0]
    g = 10
    plt.plot(0, h, 'bo')
    plt.xlabel('Time [s]')
    plt.ylabel('Height [m]')
    prev_min_max = 0
    for i in range(0, 401):
        time = (i + 1) * 0.04
        if down:
            v += g * 0.04
            h -= (v * 0.04) + (0.5 * g * 0.04 ** 2)
        else:
            v -= g * 0.04
            h += (v * 0.04) + (0.5 * g * 0.04 ** 2)

        plt.plot(time, h, 'bo')

        if h < 0:
            down = False
            h = 0
            print(format("B:\t%.2f\t%.2f\t%.2f\t%d") %
                  (v, time - prev_min_max, time, i + 1))
            prev_min_max = time
        if v < 0:
            down = True
            v = 0
            print(format("T:\t%.2f\t%.2f\t%.2f\t%d") %
                  (v, time - prev_min_max, time, i + 1))
            prev_min_max = time

    plt.show()


if __name__ == '__main__':
```

```
    user_input = getInput()
    motionSimulator(user_input)
```

Part A Sample Output

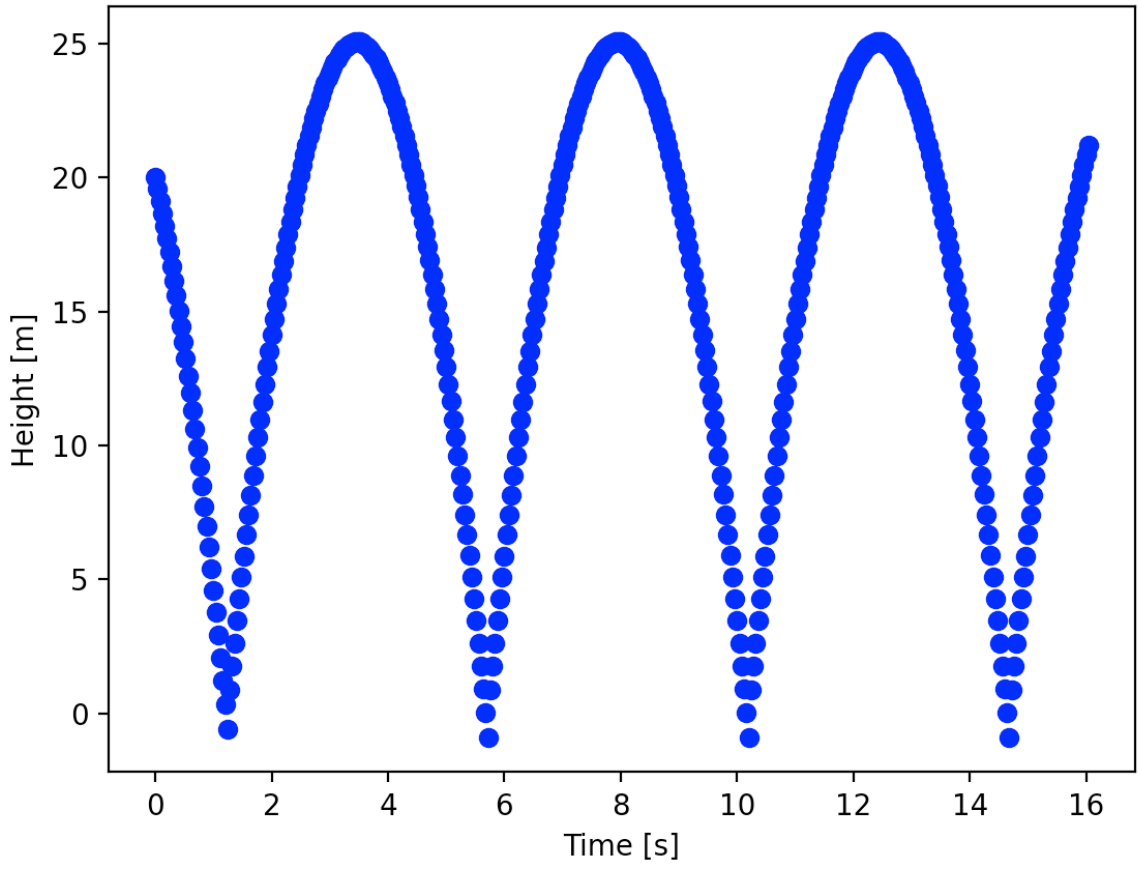
Sample 1:

```
Please enter initial height -> 20
Please enter initial speed -> 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
```



Sample 2:

```
Please enter initial height -> 20
Please enter initial speed -> 10
B:      22.40   1.24    1.24    31
T:      0.00    2.24    3.48    87
B:      22.40   2.24    5.72    143
T:      0.00    2.24    7.96    199
B:      22.40   2.24    10.20   255
T:      0.00    2.24    12.44   311
B:      22.40   2.24    14.68   367
```



Part B Source Code

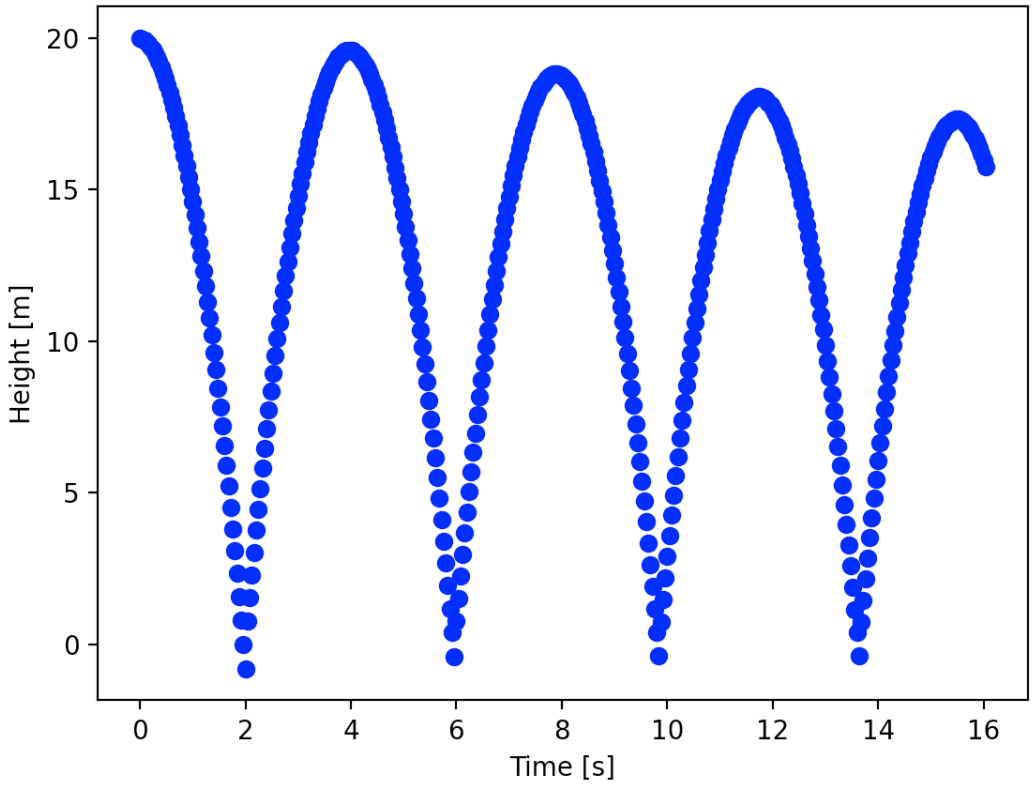
```
# Student name: Dojae Kim
# Student number: 400420323
# Student email: kim408@mcmaster.ca
# Lecture: SFWRTECH 3PR3
# Assignment 3 Part B

from typing import Tuple
import matplotlib.pyplot as plt


def getInput() -> Tuple[float, float, float]:
    """
    Get input from user - tuple of 2 floats, 1st one is initial height,
    2nd one is initial speed and 3rd one is energy loss (>= 0 and <=1)
    :return: Tuple of initial values
    """
    done = False
    init_height = 0.0
    init_speed = 0.0
    ep = 0.0
    while not done:
        try:
            init_height = float(input("Please enter initial height -> "))
            if init_height >= 0:
                done = True
        except ValueError:
            print("Initial height should be positive!")

    done = False
    while not done:
        try:
            init_speed = float(input("Please enter initial speed -> "))
            if init_speed >= 0:
                done = True
        except ValueError:
            print("Initial speed should be positive!")

    done = False
    while not done:
        try:
            ep = float(input("Please enter energy loss (from 0 to 1) -> "))
            if 0 <= ep <= 1:
                done = True
            else:
                print("Energy loss should be between 0 and 1")
        except ValueError:
            print("Energy loss should be positive!")

    return init_height, init_speed, ep


def energyLoss(v: float, ep: float) -> float:
    """
    Calculate energy loss
```

```
    :param v: Current speed
    :param ep: Energy loss
    :return: New speed considering energy loss
    """
    return v * (1 - ep ** 2)


def motionSimulator(init_values: Tuple[float, float, float]) -> None:
    """
    Simulate motion of the ball
    :param init_values: Initial values (height and speed)
    :return: None
    """
    down = True  # Initially ball travels down
    v = init_values[1]
    h = init_values[0]
    g = 10
    plt.plot(0, h, 'bo')
    plt.xlabel('Time [s]')
    plt.ylabel('Height [m]')
    prev_min_max = 0
    for i in range(0, 401):
        time = (i + 1) * 0.04
        if down:
            v += g * 0.04
            h -= (v * 0.04) + (0.5 * g * 0.04 ** 2)
        else:
            v -= g * 0.04
            h += (v * 0.04) + (0.5 * g * 0.04 ** 2)

        plt.plot(time, h, 'bo')

        if h < 0:
            down = False
            h = 0
            print(format("B:\t%.2f\t%.2f\t%.2f\t%d") %
                  (v, time - prev_min_max, time, i + 1))
            prev_min_max = time
            v = energyLoss(v, init_values[2])
        if v < 0:
            down = True
            v = 0
            print(format("T:\t%.2f\t%.2f\t%.2f\t%d") %
                  (v, time - prev_min_max, time, i + 1))
            prev_min_max = time

    plt.show()


if __name__ == '__main__':
    user_input = getInput()
    motionSimulator(user_input)
```

Part B Sample Output

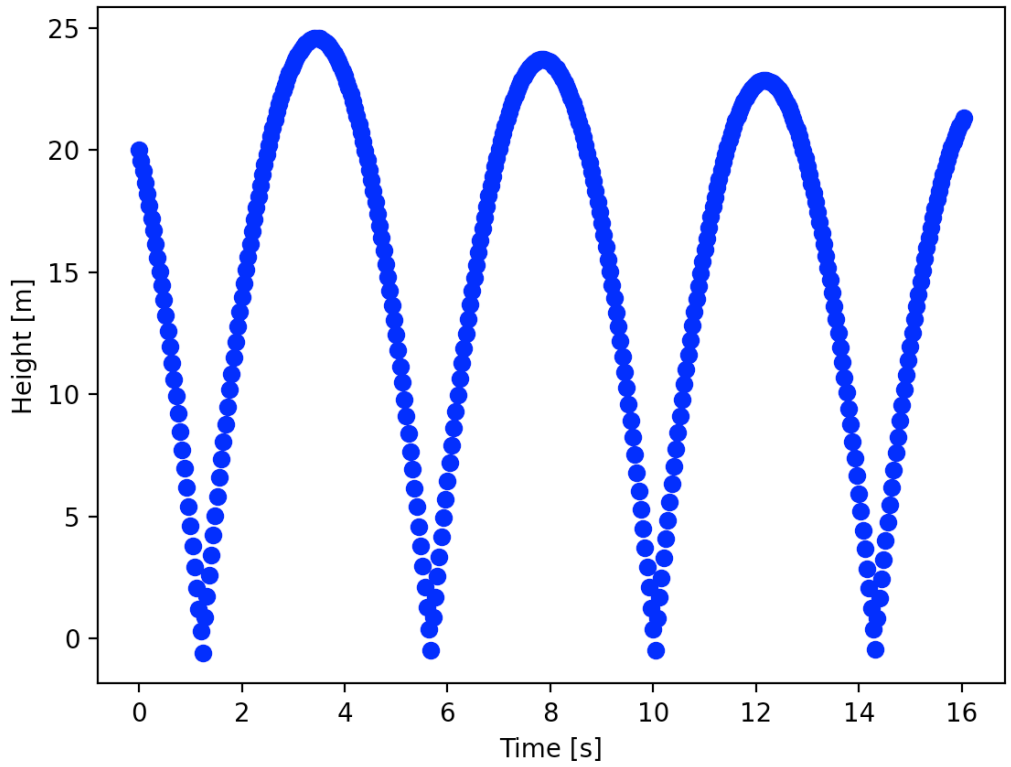
Sample 1:

```
Please enter initial height -> 20
Please enter initial speed -> 0
Please enter energy loss (from 0 to 1) -> 0.1
B:      20.00   2.00    2.00    50
T:      0.00    2.00    4.00    100
B:      19.60   1.96    5.96    149
T:      0.00    1.96    7.92    198
B:      19.20   1.92    9.84    246
T:      0.00    1.92    11.76   294
B:      18.80   1.88    13.64   341
T:      0.00    1.88    15.52   388
```



Sample 2:

```
Please enter initial height -> 20
Please enter initial speed -> 10
Please enter energy loss (from 0 to 1) -> 0.1
B:      22.40   1.24    1.24    31
T:      0.00    2.24    3.48    87
B:      22.00   2.20    5.68    142
T:      0.00    2.20    7.88    197
B:      21.60   2.16    10.04   251
T:      0.00    2.16    12.20   305
B:      21.20   2.12    14.32   358
```



Sample 3:

```
Please enter initial height -> 20
Please enter initial speed -> 20
Please enter energy loss (from 0 to 1) -> 0.3
B:      28.40   0.84    0.84    21
T:      0.00    2.60    3.44    86
B:      25.60   2.56    6.00    150
T:      0.00    2.36    8.36    209
B:      23.20   2.32    10.68   267
T:      0.00    2.12    12.80   320
B:      20.80   2.08    14.88   372
```