Lab Section:
TA Name: Student Name:

Links to the (youtube) instructional videos are at the bottom of the web page:
http://www.impactstem.org/mechanics/lab1-hw

Part 2.1: (after watching Video 2.1)

Step 1: Type the command to store the sum of 8 and 4 into a value-holder called "velocity" into the black interactive window. Then type another command to confirm that the value-holder "velocity" contains the correct value.

Below, write the commands you typed to do this.

Step 2: Type a command that will increase the value stored within value-holder "velocity" by four. Then type another command to verify that "velocity" contains the correct value.

Below, write the commands you typed to do this.

Part 2.2 (after watching Video 2.2)

Step 1: Use the following commands to plot a two points

```
> dot(0,0,red)
> dot(100,50, blue)
```

Step 2: Use the locations where the red and blue points appear to determine

- coordinates of the graph pane's corners
    - Coordinate of upper left corner: ___________
    - Coordinate of lower left corner: _________
    - Coordinate of upper right corner: __________
    - Coordinate of lower right corner: __________
- interval between tic marks: ____________

Notes on how you figured this out:

Part 2.3 (after watching Video 2.3)

Step 1. Type the following commands into the upper (white) **program window,** and then save it to your flash drive within a new folder named "program" with the name "line1.py". Confirm that you typed it in correctly by loading (and saving) it. (Hint: Use the "load" button)

```
col = 30
while col < 71:
    dot(col, 30, green)
    col = col + 1
```

Notes:

- The first two lines should **not** be indented.
- Remember to end the `while` statement with a colon (:), not a semicolon (;)
- Indent the two statements within the `while` (the last two statements) using the same number of spaces (try pressing the tab key).
- You can use the `clear()` command to erase the graphics screen (try it!).

Step 2: Save line1.py to a new name **line1a.py** (thus making a copy). Modify the program to draw a pink line segment from column 30 to column 70 in a lower row.

Hints:

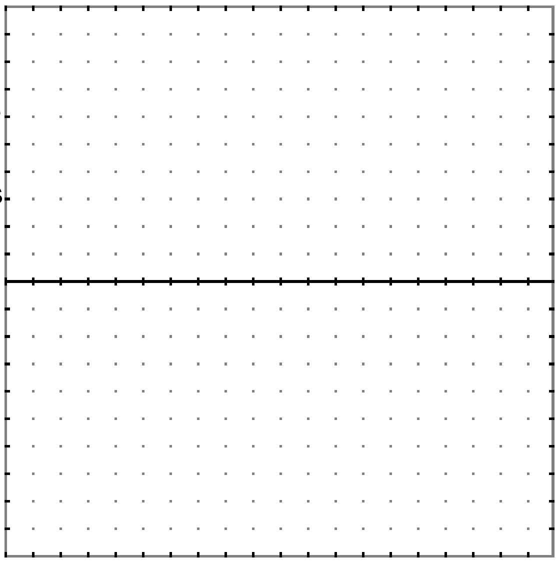
- You should only need to change one number to change the row.
- If you make a mistake, just clear() the window, re-open line1.py, save it to line2.py (over-writing your mistakes), and start over!

Once you succeed, indicate the changes you made and the principles underlying why they worked below:

Step 3:

- Make another copy of line1.py named line1b.py by
  - Opening line1.py again.
  - Saving it to line1b.py
- Modify line1b.py to draw a black horizontal line that starts (at some column you choose) to the left of column 30 and ends (at some column you choose) between 40 and 60.. Once you succeed, indicate your changes and the principles underlying why they worked below:

Part 2.4 (after watching video 2.4)



Step 1: Without running line2.py (below), indicate what you expect it to draw on the graph to the right. Be sure to label any interesting coordinates (e.g. start and end points)

```
col = 30
row = 50
while col < 71:
    dot(col, row, green)
    col = col + 1
    row = row + -1
```

Step 2: Save “line2.py” into your folder “Program” and load it. If you predicted incorrectly, please explain your confusion (full credit will be awarded for a clear explanation)

Step 3: What is the slope of the shape plotted by line2.py? Why?

Step 4: Make a copy of line2.py named line2b.py. Change the amount added to row so that the resulting graph has an upward slope. Once you have succeeded, indicate your change, the resulting slope, how they are related, and why.

Extra Credit

Part 2.5 (after watching video 2-5)

Step 1: Copy line2.py to line3.py, edit line3.py to correspond to the program at the end of video 2-5 (see below), and confirm that it works correctly. Differences with line2.py are highlighted in **bold.**

```
time = 30
pos = 50
velocity = -1
while time < 71:
    dot(time, pos, red)
    time = time + 1
    pos = pos + velocity*1
```

Step 2: Analytically compute (using your knowledge of physics and algebra) for the final values of pos and velocity (show your work below). Use the interactive window to determine the program's final values for pos and velocity. These values should match your analytically derived values. If not, figure out why and indicate what you learned.

Part 2.6 (after watching video 2-6)

Step 1. Copy line3.py to line4.py, edit it to match the following, and confirm that it functions correctly:

```
clear()
pos = 2.4       # meters
time = 2        # seconds
deltaT = 0.1    # seconds, interval
velocity = 0.1  # meters / second
while time <= 3.8:
    dot(time*20,pos*20, green)
    time = time + deltaT
    pos = pos + velocity*deltaT
```

Step 2. Copy line4.py to line4b.py and modify to represent the graph of a cart that starts its movement at 50 centimiters from the sensor and a velocity of .3 meters per second away from it, and stops 3 seconds after the movement began. Confirm that it functions correctly.

Step 3. Analytically compute (using your knowledge of physics and algebra) for the final values of pos and velocity (show your work below). Use the interactive window to determine the program's final values for pos and velocity. These values should correspond to your analytically derived values. If not, figure out why and indicate what you learned.

Part 2.7 (after watching video 2-7)

Step 1. Copy line4.py to line5.py, edit it to match the following, and confirm that it functions correctly:

```
clear()
time = 0.234      # seconds
deltaT = 0.01      # seconds, interval
pos = 1.283       # meters
velocity = -0.67   # meters/seconds
while time < 10:
    dot(time*20, pos*20, red)
    dot(time*20, velocity*20, green)
    time = time + deltaT
    pos = pos + velocity*deltaT
    if pos < .544 or pos > 2.44:
        velocity = -velocity
```

Step 2. Change in line5.py the deltaT to be 0.1 and again to be 0.001 and compare the results.