ENGR 102 – Fall 2021 - ELMS
Lab Assignment #2b

Deliverables.

Lab Assignment #2b consists of three individual activities. Please submit the following files to Mimir.

- `Lab2b_Act1.py`
- `Lab2b_Act2a.py`
- `Lab2b_Act2b.py`
- `Lab2b_Act2c.py`
- `Lab2b_Act3.py`

Activity #1: Writing Programs – individual

Convert your program from Lab Assignment 1b Activity #2 to a new program named Lab2b_Act1.py that produces identical output. However, for all of the calculations, you are to instead create variables for all values that are either constants or are values that might vary in the calculation.

Recall the following example from Lab 1b:

New code:

```
# Calculate/print area of rectangle of length 5 in and height 3 in
print("Area of rectangle is", 5 * 3, "in^2")
```

New output:

```
Area of rectangle is 15 in^2
```

Now we know about variables and assignments statements, so let's modify the code as follows:

Newer code:

```
# Calculate/print area of rectangle of length 5 in and height 3 in
length = 5 # inches
height = 3 # inches
area = length * height  # in^2
print("Area of rectangle is", area, "in^2")
```

Newer output:

```
Area of rectangle is 15 in^2
```

Please note the following:

1. You should pick **good** names for your variables.
2. You do not have to perform the entire computation in one line; you can use multiple lines to perform the computation if you want.
3. It is OK to introduce variables to hold values that are not a "final" value. For example, if you were computing the area of a circle, you might store the radius in one variable, then the radius squared in another variable, and then later multiply that by pi to compute the area.

Produce output for the following calculations:

A) Calculate the force in Newtons (N) applied to an object with a mass of 2 kg and an acceleration of 5 m/s^2. According to **Newton's Second Law** the net force applied to an object produces a proportional acceleration.

B) Calculate the wavelength of X-rays scattering from a crystal lattice with a distance between crystal layers of 0.025 nm, scattering angle of 25 degrees, and first order diffraction. **Bragg's Law** describes the scattering of waves from a crystal using the equation:

$$n * \lambda = 2 * \ d * sin\, \theta$$

where,
n is the order of diffraction
λ is the wavelength of the X-rays in nm
d is the distance between consecutive layers of atoms in nm
θ is the scattering angle

C) The standard unit of wavelength in the SI system is nanometers (nm). Calculate how much Radon-222 is left after 5 days of radioactive decay given an initial amount of 3 g and a half-life of 3.8 days. The equation for **radioactive decay** is

$$N(t) = N_0 2^{-t/t_{1/2}}$$

where,
N_0 is the initial amount in grams
t is the time in days
$t_{1/2}$ is the half-life in days

D) Calculate the pressure in kPa of 5 moles of an ideal gas with a volume of 0.15 m^3, and temperature of 425 K. The **Ideal Gas Law** is the equation of state of a hypothetical ideal gas and is a good approximation of the behavior of gases under many conditions. Use a value of 8.314 m^3 Pa/K·mol for the gas constant.

Example output:

```
Force is 10 N
Wavelength is 0.021130913087034974 nm
Radon-222 left is 1.2051168398503236 g
Pressure is 117.78166666666667 kPa
```

<u>Activity #2</u>: More linear interpolation – individual

In the team lab, your team put together a program that interpolated between two position values based on the time values when each position was observed. This was a one-dimensional (1D) interpolation, since you were interpolating only a single value, the distance traveled by the ISS. You are now going to extend that program to one that will linearly interpolate between two points in 3D.

Let's say we are tracking the change of a satellite's position with time. So, at time t1 the position is (x1, y1, z1) and at time t2 the position is (x2, y2, z2). What is the position (x0, y0, z0) at some time t0 between t1 and t2?

Refer again to the Linear Interpolation material associated with Lab Assignment 2a. That material describes the development of the equation representing linear interpolation of a dependent variable y versus an independent variable x. For the current problem, what varies linearly with what? What are the dependent variable(s)? What are the independent variable(s)?

Let’s assume that each of the position variables (x, y, z) varies linearly with time (t). Therefore, time (t) is the independent variable in each case. This means we can perform linear interpolation *three separate times* to get what we need. This can be done in three steps: 1) linearly interpolate between (t1, x1) and (t2, x2) for t0 with x0 as the result; 2) repeat for (t1, y1) and (t2, y2) for t0 with y0 as the result; 3) repeat for (t1, z1) and (t2, z2) for t0 with z0 as the result. The result will be (x0, y0, z0) associated with time t0.

a) Write a program named Lab2b_Act2a.py that will take two observed 3D positions at two points in time, and then will calculate the 3D position at a third point in time. Let’s consider only times between the two observed times. You should output the x, y, and z values for that position on separate lines. Begin by identifying the variables you will use, the names for those variables, and the computations that should occur for those variables. Then, write a program that will output the 3D position of the interpolated point on 3 separate lines.

   For this initial program, use the following data values:

   - At time 12 seconds, observed position was (2, 3, 7) meters
   - At time 85 seconds, observed position was (25, -5, 11) meters
   - You want to find the position at time 45 seconds

   Example output:

```
At time 45 seconds:
x0 = 12.397260273972602 m
y0 = -0.6164383561643834 m
z0 = 8.808219178082192 m
```

b) Now, copy Lab2b_Act2**a**.py from above into a new program named Lab2b_Act2**b**.py. Modify your program in the following ways:

   i. When printing the position, follow the output by a line of dashes (“-------------------”).

   ii. Instead of just computing the interpolation at one point and printing the result, you will now compute it at 5 points. Copy the portion of the code (copy and paste the code) that is needed to recompute interpolation 5 times. You should now interpolate at times in increments of 1 second, starting at time 45 seconds (i.e. at times 45, 46, 47, 48, 49), printing the result each time. The line of dashes will separate each computation. (Note: later we will see how we can do this more efficiently, without copying-and-pasting code, but for now, copy-and-paste your code.)

Example output (first two times only):

```
At time 45 seconds:
x0 = 12.397260273972602 m
y0 = -0.6164383561643834 m
z0 = 8.808219178082192 m
----------------------
At time 46 seconds:
x0 = 12.712328767123287 m
y0 = -0.7260273972602738 m
z0 = 8.863013698630137 m
```

c) Finally, copy Lab2b_Act2**b**.py into a new program, Lab2b_Act2**c**.py. Modify your program in the following way:

i. Create variables for the **starting** time of interpolation, and the **ending** time of interpolation.

ii. You should display the results from interpolating at 5 points, **evenly spaced** from the beginning time to the ending time, inclusive.

iii. Experiment on your own with assigning different values to those variables and verify that you are in fact interpolating correctly from one point to another.

iv. For the final version that you submit, show an interpolation from time 30 to 60 seconds.

Example output (first two times only):

```
At time 30 seconds:
x0 = 7.671232876712328 m
y0 = 1.0273972602739727 m
z0 = 7.986301369863014 m
----------------------
At time 37.5 seconds:
x0 = 10.034246575342465 m
y0 = 0.2054794520547949 m
z0 = 8.397260273972602 m
```

Activity #3: Problem solving – individual

Create a program named Lab2b_Act3.py consisting of **only** the following lines of code to produce the output shown below. You may put these lines of code in any order and can re-use the lines as much as you want. There is more than one way to achieve the result – try to see if you can obtain the output using fewer lines of code. Hint: you can only print `z` to the screen, so you have to build the value of `z` that you want using the other statements, then print `z`.

```
x = 1
y = 10
z = 0
x = y
x += 1
y += x
y *= x
z += x
z += y
print(z)
```

You may also use blank lines and single line # comments in your code (including the header), **everything else will be marked as "not allowed"**

For example, say that you wanted to print the number "1" to the screen. The following lines would do the trick:

```
z = 0
x = 1
z += x
print(z)
```

Your program should print out the following, when run:

```
1
25
102
1000000000
2468
```

[Note: that's 10^9]