Attach the assignment to your HW/Lab | ENGR 102 Section **Lab 2b**
Date: _____ | **DUE DATE: see webpage**
You name ____

ENGR 102 Sect #____ Lab 2b

100 points

Reading assignment:

Lecture Slides	**L01- L02**
zyBook	*Ch1 -2*

Attention!!

For submission: pdf/word file and all py-files as asked in the assignment. If you do pictures by the phone –please make sure that we can read them. Do not submit multiple picturess, collect them all into one file (word or pdf). . You will be allowed to resubmit and reupload HW as many times as you want to within the due date/time, only last submission will be graded. No late submissions.

For submission you may use this file as a template: rename file including your name. Do not forget to put your name inside of this file as well.

If you are submitting py- files, make sure that they have a header. For this submission use Individual Header. Submit individually.

Additional instructions:

- Please include **individual header information (include your name and session number)** at the top of each file.
- Please submit all files on the list as a single submission on Canvas. You may practice on ZyBook.To do that read instructions on ZyBook carefully how to name files.
- Please do not zip your files.
- In the File "LabXX results_YourlastName.pdf" you need to put for each problem
  - Brief problem statement of the problem
  - Copy of your code
  - Screenshots of your output results

Assignment: Writing your Programs – To Do individually, in lab or outside

You are to write the following 3 programs (though one of them has 2 versions), each of which should be done individually. Once the programs are written, you should submit 4 programs and a result file on Canvas.

- Lab2b_Prog1_name.py
- Lab2b_Prog2a_name.py
- Lab2b_Prog2b_name.py
- Lab2b_Prog3_name.py

+ Summary result file

Program 1: [25 points]
Please name this program Lab2b_Prog1_name.py.

Convert your program from Lab Assignment 1b Activity #2 to a new program named Lab2b_Prog1_name.py. You are to convert that program to a new program 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. As an example, if you previously had a line like:

```
print(3+2)
```

you would want to change it to a sequence of assignments using variables such as:

```
a = 3
b = 2
c = a+b
print(c)
```

Please note the following:

a) Your print statements should each print just a **single variable**. <u>You are not performing calculations inside of the print statement</u>
b) You should pick good names for your variables.
c) You do not have to perform the entire computation in one line; you can use multiple lines to perform the computation if you want.
d) 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.

For example:
Recall the following example from LabAssignment1b:

<u>Last week code:</u>

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

<u>last week output</u>: `Area of rectangle is 15 in^2`

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

<u>Newer code:</u>

```
# 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")  # area of rectangle in in^2
print()
```

<u>Newer output:</u> `Area of rectangle is 15 in^2`

<u>Program 1 Requirements:</u>

This program should produce several lines of output as described in parts a) through f) below. Repeat the steps from LabAssignment1b, but use variables and assignment statements to add functionality to the code. And now you can have only 1 py-file or several like last time. For each of the equations provide comments in the code to document the name of the equation and the name of input variables and the corresponding units. Also, print the results to the screen formatted similarly to the example above including numerical results, units, and formatting.

As a reminder, your program was to print the following lines (the first 2 don't change from before):

a) [5] Your name, UIN, and section number of ENGR 102 that you are enrolled in
Summary of the assignment: for example assignment number, or "this program calculates.."

What do you need to calculate?
(You may choose to put code/output for each task separately or all together, but in that case, you need to put a comment which line on your program calculates what.)

b) [5] Calculate the force in Newtons applied to an object with mass 3 kg and acceleration 5.5 m/s^2. According to Newton's Second Law the net force applied to an object produces a proportional acceleration.

Put your code/output here:

c) [5] 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

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

The standard unit of wavelength in the SI system is nanometers (nm).

Put your code/output here:

d) [5] Calculate how much Radon-222 is left after 3 days of radioactive decay given an initial amount of 5 g and a half-life of 3.8 days. The equation for radioactive decay is

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

Put your code/output here:

a) [5]] Calculate the pressure of 5 moles of an ideal gas with a volume of 0.25 m^3, and temperature of 415 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^3Pa/K·mol for the gas constant. The standard unit of pressure in the SI system is kilopascals (kPa).

Put your code/output here:

Example output:

```
Force is 16.5 N
Wavelength is 0.021130913087034974 nm
Radon-222 left is 2.8927755932067996 g
Pressure is 69.00619999999999 kPa
```

You name ____

Program 2 [2x25=50 points]: More linear interpolation – individual

Lab2b_Prog2_name.py

The purpose of this activity is to practice writing simple programs that require multiple variables and to ensure you understand the idea of interpolation.

You are to write a short program that performs the linear interpolation. Here is the scenario.

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 (t0, x0) and (t2, x2) for t1 with x1 as the result; 2) repeat for (t0, y0) and (t2, y2) for t1 with y1 as the result; 3) repeat for (t0, z0) and (t2, z2) for t1 with z1 as the result. The result will be (x1, y1, z1) associated with time t1.

Task 2a [25]

a) Write a program named Lab2b_Prog2a_name.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 (8, 6, 7) meters
   - At time 85 seconds, observed position was (-5, 30 9) meters
   - You want to find the position at time 30 seconds

   Example output:

```
At time 30.0 seconds:
x1 = 4.794520547945206 m
y1 = 11.917808219178081 m
z1 = 7.493150684931507 m
```

Do not forget comments. Points will be deducted if you don't have them in your program and in your outputs.

Put your code/output here:

You name ____

Task 2b [25 points]

Now it is time to code. Create py-file, name, think of the mane of the file to make it clear what lab you are working on. For example, Lab2b_Prog2b_name.py

You can copy Lab2b_Prog2**a**.py from above into a new program named Lab2b_Prog2**b**.py. Modify your program in the following ways:

i. [5] When printing the position, follow the output by a line of dashes ("-------------------").
ii. [7] Instead of just computing the interpolation at one point and printing the result, you will now compute it at 5 points. You may copy and paste the portion of your code that is needed to recompute interpolation 5 times.
iii. [5] Create variables for the **starting** time of interpolation, and the **ending** time of interpolation. Display the results from interpolating at 5 points, **evenly spaced** from the beginning time to the ending time, inclusive.
iv. [8] Interpolate, starting at time 30 seconds and ending at time 60 seconds, 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 30.0 seconds:
x1 = 4.794520547945206 m
y1 = 11.917808219178081 m
z1 = 7.493150684931507 m
-----------------------
At time 37.5 seconds:
x2 = 3.4589041095890414 m
y2 = 14.383561643835616 m
z2 = 7.698630136986301 m
```

Do not forget comments. Points will be deducted if you don't have them in your program and in your outputs.

In the comments of your code put

1. The header
2. Summary of the problem statement. What does it do?
3. Your derived Formula
4. Use comments to make a nice structure of your code.
5. You may comment used variables

Put your code/output here:

You name ____

Task 3 [25 points]

Python code soup – individual

Create a program named python_code_soup.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 # style 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
26
102
1000000000
8675
```

[Note: that's 10^9]

Put your code/output here:

Turn in All your programs and completed word/pdf file with comments derivations and screenshots on Canvas