Attach the assignment to your HW/Lab ENGR 102 Section **Lab 7b**
Date: _____ **DUE DATE: 10/10/2022**
You name _____Team # (table)_____

ENGR 102 Sect Lab 7b- indiv
Canvas part
50 points- Canvas submission
34 points = 50% Zybook

Reading assignment:

Lecture Slides	**L07**
zyBook chapter 7	**Chapter 7**
Halterman python book handout	**Chapter 9 Lists**

Attention!!

Individula submission

Submit your Py-files together with your word/pdf file. In your report file put screenshots of your programs and outputs. Also include any derivations, comments and supplemental notes in your word/pdf files.

No pictures by the phone – it is impossible to read. 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. For submission you may use this file as template: rename file including your name. Do not forget to put your name inside of this file as well.

EXTRA HANDOUTS ARE AVAILABLE ON CANVAS

Problem 1 [10 points]

Create a list of integer numbers using **the range function** to produce a regular sequence of integers. Ask the user to put the minimum number of the list, the maximum number of the list and step. You should ask user if he/she wants to create another list. Keep creating lists until the user wants to stop. You can do it for example by analyzing user's response Yes and No. As the output print your created lists and rules by what you created them. Put your screenshots here.

Problem 2 [8 points]

For a given list ask user if he/she wants to create a new list that repeats n-times itself. If Yes- create a new list that repeats itself n-times, (n is user specified). Put your screen shots here.
For example of input "Howdy" and 3:
The output is HowdyHowdyHowdy

Attach the assignment to your HW/Lab ENGR 102 Section **Lab 7b**
Date: _____ **DUE DATE: 10/10/2022**
You name _____Team # (table)_____

Problem 3 [22 points]

Complete the following table by supplying the command and m and n values in **the slice assignment a[m:n]** or **list** statement needed to produce the indicated list from the given original list. You may use other commands as well to create the desired output.

	Original list	Target List	Slice command (may vary) Do not do loops. Use slicing
1	[2, 4, 6, 8, 10]	[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]	
2	[2, 4, 6, 8, 10]	[-10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10]	
3	[2, 4, 6, 8, 10]	[2, 3, 4, 5, 6, 7, 8, 10]	
4	[2, 4, 6, 8, 10]	[2, 4, 6, 'a', 'b', 'c', 8, 10]	
5	[2, 4, 6, 8, 10]	[2, 4, 6, 8, 10]	
6	[2, 4, 6, 8, 10]	[]	
7	[2, 4, 6, 8, 10]	[10, 8, 6, 4, 2]	
8	[2, 4, 6, 8, 10]	[2, 4, 6]	
9	[2, 4, 6, 8, 10]	[6, 8, 10]	
10	[2, 4, 6, 8, 10]	[2, 10]	
11	[2, 4, 6, 8, 10]	[4, 6, 8]	

Problem 4 [10]

Run 2 examples from Richard L. Halterman "LEARNING TO PROGRAM WITH PYTHON"

- P89, chapter 5.4 Example 1 "Print multiplication table" [20points]
- P91 chapter 5.4 Example "Permutation of ABC" [20 points]

Understand the algorithms

Submit your screenshots here and py files on canvas

Attach the assignment to your HW/Lab ENGR 102 Section **Lab 7b**
Date: _____ **DUE DATE: 10/10/2022**
You name _____Team # (table)_____

This is a second part of the HW 7b _ind, please submit o n ZyBook

Lab: Topic 7 (individual)
34 pts =50%

Deliverables:

There are four deliverables for this individual assignment. Please submit the following files to zyBooks:

- `pig_latin.py` 7.25 [8 pts]
- `leet_speak.py` 7.26 [5 pts]
- `vector_math.py` 7.27 [9 pts]
- `kaprekars_constant.py` 7.28 [12 pts]
- `kaprekars_challenge.py` (optional) 7.29 [2pts]

Activity #1: Pig Latin (https://en.wikipedia.org/wiki/Pig_Latin) – individual

This program is meant to help give you practice with string manipulation. "Pig Latin" is a way of converting words in standard English to similar words that sound different. The rules for converting from standard words to Pig Latin are as follows:

- If a word starts with a consonant (or cluster of consonants that form one sound), move the consonant(s) to the end of the word, and add "ay" to the end
  - Example: "computer" becomes "omputercay"
- If a word starts with a vowel, add "yay" on to the end of the word
  - Example: "engineering" becomes "engineeringyay"
- Note: treat "y" as a vowel for this assignment

Write a program named pig_latin.py that takes as input from the user one string containing multiple words each separated by a single space. Have your program convert the words to Pig Latin, then print both the original words and Pig Latin versions using the format shown below. User input is shown in bold and red text. Note: the user may enter an arbitrary number of words.

Example output (using input **howdy aggies whoop**):

```
Enter word(s) to convert to Pig Latin: howdy aggies whoop
In Pig Latin, "howdy aggies whoop" is: owdyhay aggiesyay oopwhay
```

Activity #2: Leet speak – individual

1337 (or leet) is an alternative alphabet used mostly on the internet that replaces certain letters with other characters, such as numbers. Write a program named leet_speak.py that takes as input from the user a string of text, converts the words to leet, and prints the converted text. Your program must use a dictionary.

Use the following letter to number conversions in your program: a > 4, e > 3, o > 0, s > 5, t > 7

Example output (using input **howdy aggies whoop**):

```
Enter some text: howdy aggies whoop
In leet speak, "howdy aggies whoop" is:
h0wdy 4ggi35 wh00p
```

Activity #3: Vector math – individual

This program is meant to give you practice with lists and looping on them, as well as practice with vector computations. Write a program named vector_math.py that lets a user enter two vectors, A and B, of the same arbitrary length and store each vector as a Python list. Have the user enter each vector in one line with spaces between elements. Then, you should output the results of these computations using the format shown below. Print any resulting lists to include `[]` and `,`.

- The magnitude of vector A and the magnitude of vector B printed to five (5) decimal places
- A + B
- A – B
- The dot product (inner product) of A and B printed to two (2) decimal places

Note: You should use **lists** (and loops) when solving this problem. You may **NOT** use sympy or numpy. Later we will see some other ways to work with vectors more directly.

Example output using vectors A = [1, 2, 3] and B = [4, 5, 6]:

```
Enter the elements for vector A: 1 2 3
Enter the elements for vector B: 4 5 6
The magnitude of vector A is 3.74166
The magnitude of vector B is 8.77496
A + B is [5.0, 7.0, 9.0]
A - B is [-3.0, -3.0, -3.0]
The dot product is 32.00
```

Activity #4: Kaprekar's Constant – individual

6174 is known as Kaprekar's Constant (https://en.wikipedia.org/wiki/6174_(number)) after the Indian mathematician D. R. Kaprekar. This number can be obtained using Kaprekar's routine:

1. Take any four-digit number that has at least two different digits (add leading zeros to numbers with fewer than four digits)
2. Sort the digits in descending and then in ascending order to get two four-digit numbers, adding leading zeros if necessary
3. Subtract the smaller number from the bigger number to get a new four-digit number
4. Go back to step 2 and repeat

Kaprekar's routine will always reach its fixed point, 6174, in at most 8 iterations. Once 6174 is reached, the process will continue to yield 7641 – 1467 = 6174.

For example, choose 3524:
5432 – 2345 = 3087
8730 – 0378 = 8352
8532 – 2358 = **6174**

For example, choose 137 (add a leading zero):
7310 – 0137 = 7173
7731 – 1377 = 6354
6543 – 3456 = 3087
8730 – 0378 = 8352
8532 – 2358 = **6174**

The only four-digit numbers for which Kaprekar's routine does not reach 6174 are repdigits (numbers where all digits are the same, such as 1111) which give 0000 after a single iteration. All other four-digit numbers eventually reach 6174, as long as leading zeros are used to keep the number of digits at four.

Write a program named kaprekars_constant.py that takes in an integer from the user between 0 and 9999 and implements Kaprekar's routine. Have your program output the sequence of numbers to reach 6174 and the number of iterations to get there. Format your output as shown below.

Example output (using input **2026**):

```
Enter a four-digit integer: 2026
2026 > 5994 > 5355 > 1998 > 8082 > 8532 > 6174
2026 reaches 6174 via Kaprekar's routine in 6 iterations
```

Hints:

- Consider converting numbers to strings for sorting, and back to numbers for calculating
- Pad the number with 0s when it has fewer than four digits (remember string concatenation?)
- The `list()`, `<listname>.sort()`, and `<string>.join()` methods may be helpful

Activity #4 challenge: More Kaprekar's Constant (optional 2 bonus points)

Modify your program from Activity 4 to compute the sum of the number of iterations required to reach 6174 (or 0000) using Kaprekar's routine for all four-digit numbers, from 0000 to 9999. Name your file kaprekars_challenge.py. If your program calculates the total number of iterations correctly, you will receive 2 bonus points on this assignment.

Example output:

```
Kaprekar's routine takes ????? total iterations for all four-digit numbers
```