COSC 130 – HW 02 Instructions

General Instructions

Create a new notebook named **`HW_02_YourLastName.ipynb`** and complete problems 1 – 8 described below.

Any set of instructions you see in this document with an orange bar to the left will indicate a place where you should create a markdown cell. For each new problem, create a markdown cell that indicates the title of that problem as a level 2 header.

Any set of instructions you see with a blue bar to the left will provide instructions for creating a single code cell.

Read the instructions for each problem carefully. Each problem is worth 6 points. An additional 2 points are allocated for formatting and following general instructions.

The assignment should be completed using only base Python commands. Do not use any external packages.

Assignment Header

Create a markdown cell with a level 1 header that reads: "COSC 130 - Homework 02". Add your name below that as a level 3 header.

Problem 1: Altering Lists

In this problem, you will create an update lists containing information about employees within a particular department at a company. The table below provides the names of the employees in the department, as well as the number of years of service they have at the company:

Employee	Years of Service
Alex	5
Beth	15
Chad	2
Drew	8
Emma	4
Fred	11

Create two lists. One should be named **`employees`** and should contain the names of the employees as strings. The second should be named **`years`** and should contain the number of years of service for each employee.

The lists should be created in "parallel" so that the values in the two lists at a particular index refer to the same person. Furthermore, the lists should be ordered in decreasing order of service. In other words, the person with the greatest number of years of service should appear first in the list, and the person with the fewest years of service should appear last in the list.)

You do not need to print anything in this cell.

Suppose that two employees from other departments transfer to this department. The new members are named Gina and Herb. Gina has 12 years of service at the company and Herb has 3 years of experience.

In addition, the company hire a new employee, Iris, to join the department. Iris has 0 years of experience with the company.

Update **`employees`** and **`years`** to include these new members. The lists should still be ordered by years of service, in decreasing order.

Print both lists.

Fred accepts a position at another company. Remove his information from the lists.

It is discovered that due to a clerical error, Emma's years of service was recorded incorrectly. She actually has 6 years of experience with the company rather than 4. Update the lists to reflect this new information. The lists should again be ordered by years of service, in decreasing order.

Print both lists.

Problem 2: Sorting and Slicing Lists

In this problem, you will get practice with sorting and slicing lists, and with using list functions such as **`max()`** and **`min()`**.

Create a code cell and copy the following code into it (without the indentations).

```
import random
random.seed(12)
random_list = random.sample(range(500), 99)
```

This code will create a list named **`random_list`** that contains 99 randomly generated integers. We will discuss how this code works later in the course.

Print the length of **`random_list`**. Then print the first 10 elements of this list, all on the same line. Format your output as follows:

```
Length of random_list: xxxx
First 10 Elements of random_list: xxxx
```

Use the **`sum()`** function to calculate the sum of the elements in random_list. Print the result in the format shown below. Match the formatting exactly, including the period at the end of the sentence. There should be no space between the sum and the period.

```
The sum of the elements in random list is xxxx.
```

Create a list named **`sorted_list`** that contains the same elements of **`random_list`**, but sorted in increasing order. The order of the elements in **`random_list`** should not be changed. Print the first ten elements of both lists on separate lines. Format your output as follows:

```
First 10 Elements of sorted_list: xxxx
First 10 Elements of random_list: xxxx
```

Create lists named **`bot_slice`**, **`mid_slice`**, and **`top_slice`** as follows:

- **`bot_slice`** should contain the smallest 33 elements in **`sorted_list`**.
- **`mid_slice`** should contain the middle 33 elements in **`sorted_list`**.
- **`top_slice`** should contain the largest 33 elements in **`sorted_list`**.

Print the six messages shown below, with the appropriate numerical values inserted in place of the xxxx symbols. Include the periods at the end of the sentences, and include blank lines between each pair of sentences, as shown.

```
The smallest element of bot_slice is xxxx.
The largest element of bot_slice is xxxx.

The smallest element of mid_slice is xxxx.
The largest element of mid_slice is xxxx.

The smallest element of top_slice is xxxx.
The largest element of top_slice is xxxx.
```

Problem 3: Magnitude of a Vector

In mathematics, a vector is a quantity that conveys both a sense of direction and magnitude, or scale. Vectors are typically represented by a sequence of numbers inside of some type of brackets or parentheses. To allow us to easily think of vectors as lists, we will use square brackets to indicate a vector.

Suppose that a vector v with n elements is defined as follows: $v = [v_1, v_2, \dots, v_n]$. The **magnitude** of v is calculated by squared each element of v, summing the results, and then taking the square root of the sum. That is to say:

$$\text{magnitude of } v = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

A **unit vector** is a vector whose magnitude is equal to 1. A unit vector can be created from any vector v by dividing each element of v by the magnitude of v.

Create a list named **`vector`** to represent the following vector: `[11, 6, -8, 4, -13, 2]`. Then calculate the magnitude of **`vector`**, storing the results in a variable named **`mag_v`**. You will need to use a loop to perform this calculation. For the sake of efficient memory use, you should NOT create any additional lists to perform this calculation. Recall that you can take the square root of a number by raising it to the power of 0.5.

Print the value of **`mag_v`** rounded to 4 decimal places. The value stored in the variable should not be rounded, only the value displayed.

We will now create a unit vector from the vector we have created.

Use the values stored in **`vector`** and **`mag_v`** to calculate a unit vector, storing the results in a vector named **`unit_vector`**. You will need to use a loop or a list comprehension to perform this task.

For the purposes of display, create another list named **`rounded_unit_vector`**. The elements of this list should be calculated by rounding each of the elements of **`unit_vector`** to 4 decimal places. You can use a loop or a list comprehension for this task.

Print the list **`rounded_unit_vector`**.

We will now calculate the magnitude of the second vector we created to confirm that it is, in fact, a unit vector.

Calculate the magnitude of **`unit_vector`** using the same technique you used to calculate the magnitude of **`vector`**. Store the results in a variable named **`mag_u`**, and then print the value of this variable, without rounding it.

Problem 4: Fibonacci Sequence

The **Fibonacci sequence** is a sequence of numbers defined as follows:

- The first two terms in the sequence are 0 and 1.
- Any later terms are determined by summing together the two previous terms of the sequence.

For example, the first 7 terms of the Fibonacci sequence are: 0, 1, 1, 2, 3, 5, 8

Create a list containing the first 30 terms of the Fibonacci sequence, storing the values in a list named **`fib`**. Start by creating a list contained the first two elements: 0 and 1. Then use a loop to add the next 28 elements, one at the time.

Print the list **`fib`**.

Problem 5: Printing a Table

Consider the following table, which represents a database of (fictional) student information.

First	Last	Major	Credits	GPA
Anna	Anderson	Mathematics	110	3.56
Beatrice	Brown	Data Science	83	3.24
Charles	Clark	Biology	64	3.87
David	Daniels	Chemistry	35	2.83
Emma	Edwards	Computer Science	104	3.61

In this problem, you will be asked to create 5 parallel lists to store the contents of the columns in the table above. You will then use these lists to print a version of this table.

Create lists named **`first`**, **`last`**, **`major`**, **`credits`**, and **`gpa`** to store the contents of each of the columns in the table above. Each list should represent the column for which it is named, and the order of elements in the list should be in descending order by row. The lists **`fist`**, **`last`**, and **`major`** should contain string values, **`credit`** should contain integers, and **`gpa`** should contain floats.

Use a loop to print the contents of the lists in the format shown below. Match the format exactly, including the following the alignment of the columns and the spacing between columns. To assist with the alignment, note that the dashed line contains 56 dashes.

```
First       Last        Major              Credits    GPA
--------------------------------------------------------
Anna        Anderson    Mathematics            110   3.56
Beatrice    Brown       Data Science            83   3.24
Charles     Clark       Biology                 64   3.87
David       Daniels     Chemistry               35   2.83
Emma        Edwards     Computer Science       104   3.61
```

Problem 6: Calculating Sum of Squared Errors

Suppose that a biologist is studying the relationship between the length and weight of yellow-bellied marmots. Based on prior work, she believes that the length and weight of adult marmots can be approximately modeled by a relationship of the following form: $weight = 0.6 * length - 5.2$

To test this model, the biologist collects a sample of 20 adult marmots. The lengths and weights of the marmots are as follows:

```
length = [22.7, 22.4, 25.8, 21.3, 20.1, 22.1, 21.1, 25.3, 26.9, 26.9,
          23.0, 23.8, 26.2, 20.4, 23.0, 21.9, 23.5, 27.8, 25.3, 25.9]

weight = [9.2, 8.8, 10.7,  8.3, 6.2, 8.6, 7.2, 11.2, 10.5, 11.3,
          9.6, 9.9, 10.9,  5.9, 9.5, 9.1, 9.7, 11.6, 10.2, 10.5]
```

Create these lists, as the appear here. You do not need to print anything in this cell.

Create a list named **`pred_weight`** containing the predicted weight for each marmot, according to the biologist's model.

You should begin by creating **`pred_weight`** as an empty list. Then loop over the elements of **`length`**. Each iteration of the loop should calculate a new predicted weight using the formula provided above, and then append it to the list **`pred_weight`**.

Print **`pred_weight`**.

The biologist wishes to score her model using the sum of squared errors (**SSE**) metric. This metric is calculated as follows:

1. For each observation, calculate the difference between the true weight and the predicted weight.
2. Square this difference.
3. Sum the squared differences over all observations

Use a for loop to calculate the model's **SSE** score for this sample of marmots.

There are multiple ways to perform this task, but for the sake of efficiency, I would like for you to do it **without creating any new lists** and using **only one loop**.

Print the value of **SSE**, rounded to four decimal places. Note that the value stored in **SSE** should not be rounded, only the value displayed.

Problem 7: Determining Monthly Payments

In this problem we will be determining the monthly payment that would be required to repay a mortgage on a home in 30 years for a range of different house prices.

Assume that a loan collects interest at a monthly interest rate of `i`, and let n denote the number of monthly payments (for a 30-year loan, then $n = 360$). The size of the loan (denoted by L) and the required monthly payment (denoted by PMT) are related according to the following equation:

$$L = PMT \cdot \frac{1 - (1+i)^{-n}}{i}$$

Assume that you are considering purchasing a house and have been approved for a 30-year loan with a 0.4% monthly interest rate. We will calculate the monthly payment required for a several house prices between \$100,000 and \$200,000. Specifically, we will consider a sequence of prices starting at \$100,000 and increasing in increments of \$10,000 up to (and including) a price of \$200,000.

Perform the following steps in a single code cell.

1. Create variables named **`i`** and **`n`** to store the interest rate rate and the number of monthly payments. These values will remain constant throughout this problem.
2. Create a variable name **`start`** and set it equal to 100,000. Create another variable named **`increment`** and set it equal to 10,000.
3. Use a loop to perform the tasks below for each of the house prices that we wish to consider. Note that since we are using **`i`** to denote the interest rate, you will need to select a different name for the loop counter.
   - Calculate the new loan amount to be considered, storing the result in a variable.
   - Calculate the require monthly payment for this loan amount, rounded to 2 decimal places.
   - Print the result in the format shown below. Match the format exactly, including the inclusion of the period at the end of the sentence. There should be no spaces following the dollar signs.

     ```
     A loan of $xxxx would require monthly payments of $xxxx.
     ```

No lists should be created for this problem other than the three described in Step 1. Only one loop should be used in this problem.

Problem 8: Adding Matrices

In this problem, you will be asked to calculate the sum of two 3x5 matrices. Each matrix will be stored as a list of lists, with each sub-list representing a single row of the matrix.

Consider the following two matrices:

$$A = \begin{bmatrix} 13 & 43 & 28 & 22 & 41 \\ 17 & 39 & 46 & 16 & 21 \\ 41 & 34 & 31 & 25 & 14 \end{bmatrix} \qquad B = \begin{bmatrix} 35 & 29 & 43 & 21 & 31 \\ 48 & 26 & 19 & 17 & 23 \\ 32 & 34 & 24 & 16 & 27 \end{bmatrix}$$

Create two lists named **A** and **B** to represent the matrices above. Each of these lists should contain three lists, with each sub-list containing 5 integers representing a row in the matrix. For example, the first element of **A** should be the list **`[13, 43, 28, 22, 41]`**.

We will now use loops to calculate the sum of the matrices A and B.

Create an empty list named **`AplusB`**. Write a nested loop that adds the elements of **A** and **B**, storing the result in **`AplusB`**. Each time the outer loop executes, it should calculate a new row of the sum and should finish by adding this row to **`AplusB`**. The final structure of **`AplusB`** should be the same as **A** and **B**.

Print the rows of **`AplusB`**, with each row appearing on its own line.

Submission Instructions

When you are done, click **Kernel > Restart and Run All**. If any cell produces an error, then manually run every cell after that one, in order. Save your notebook, and then export the notebook as an HTML file. Upload the HTML file to Canvas and upload the IPYNB file to CoCalc, placing it in the **Homework/HW 02** folder.