Programming assignment #1

Course: CHE1147H - Data Mining in Engineering

1 Python data structures and other essentials

1.1 Tuples

1. Create the tuple named **apps_tuple** with the string elements "Google", "Facebook", "Amazon", "Netflix", "AirBnB", "Instagram".
2. Extract the second element by using its index **and** the last element by using its **negative** index.
3. Slice the elements from "Facebook" to "Netflix" **inclusive** on both ends.
4. Try appending the element "Messenger" to the tuple. What do you observe? Why?

1.2 Lists

1. Define the list **apps_list** with the same elements as the tuple above.
2. Slice the first three elements of the list with the **shorthand** syntax, i.e. **not** by typing explicitly all the indexes 0, 1, and 2; use the symbol **:** instead.
3. Slice all the elements after index 3 inclusively with the shorthand syntax.
4. Append the element 'Messenger' to the end of the list **and** insert the element 'Youtube' at index 1.
5. Remove element 'Facebook' and confirm with the function **in** whether 'Facebook' has been removed from the list.
6. Concatenate the last list with the list ['Linkedin', 'Twitter'].

1.3 Dicts

1. Create the dict named **apps_dict** with the same **values** as in 1.1.1. and **keys**: app0, app1, etc.
2. Access the element with key app1; then replace its value with 'Youtube'.
3. Add a new key-value pair: app6-Messenger.
4. What does the syntax: apps_dict["app1"]="Messenger" do?
5. What does the syntax: apps_dict["App1"]="Facebook" do? Why?
6. What does the syntax: del apps_dict["App1"] do?

1.4 List comprehension

1. Create the list named **values** with integers: 7, 12, 9, 18, 15.
2. Create a list comprehension that takes the object **values** and returns the square of every value.

1.5 Functions and control flow

1. Generate a sample of 10 random integer numbers between 600 and 900 (hint: use Numpy's random_integers function).
2. Create a function called **credit_score** that reads the sample and returns the output 'Low' when the input is [600, 699], 'Medium' when in [700, 799] and 'High' when in [800, 900].

2 Linear algebra in Numpy

2.1 Matrix calculations

1. Create matrices A and B as numpy arrays and calculate **2A, -3B** and **A+B**

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 2 & 3 \end{bmatrix} \qquad B = \begin{bmatrix} -4 & 1 \\ 3 & -1 \\ -2 & 1 \end{bmatrix}$$

2. Create matrices A and B as numpy arrays and calculate **A*B** and the **inverse of B**

$$A = \begin{bmatrix} 2 & -6 \\ -4 & 0 \\ 1 & 5 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix}$$

3. Replace the values of the elements with abs(value)≥3 with the value 3. Also, calculate the **determinant** of the original matrix A

$$A = \begin{bmatrix} 1 & 0 & 4 & 1 \\ -2 & 1 & -3 & 2 \\ 0 & 0 & 0 & 2 \\ 3 & 2 & 1 & -1 \end{bmatrix}$$

2.2 Norms and eigenvalues

1. Calculate the **maximum** and **Euclidean or Frobenius norm** of each of the three vectors:

$$x_1 = \begin{bmatrix} 1 & -2 & 3 \end{bmatrix}^T \qquad x_2 = \begin{bmatrix} 2 & 0 & -1 & 2 \end{bmatrix}^T \qquad x_3 = \begin{bmatrix} 0 & 1 & -4 & 2 & -1 \end{bmatrix}^T$$

2. Calculate the l_2 and l_∞ **norms** of the matrices:

$$A_1 = \begin{bmatrix} 1 & -2 \\ 4 & 3 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ -1 & 1 & 1 \end{bmatrix}$$

3. Calculate the **eigenvalues** of the matrix:

$$A = \begin{bmatrix} -2 & -2 & 3 \\ -10 & -1 & 6 \\ 10 & -2 & -9 \end{bmatrix}$$

3 Pandas manipulations

1. Import file **Gas_prices.xls** as a pandas DataFrame. Create a function that replaces NaN with the interpolation between the adjacent values (i.e. the prior and the subsequent to the NaN). Do **not** use existing interpolation functions, built your own.

2. Find the index, the date and the value of the **min** and **max** gas price.

3. Calculate the following descriptive statistics: mean, median, quantile, skewness and kurtosis.

4. Create two new columns with the Month and Year. Pivot the table by Month(index) and Year(columns). Calculate and plot the monthly average gas price. This should give you one line per year.

5. Create a new column with the **season** of the year, set it as a secondary index and calculate the average of every season (regardless of the year) to create the following table:

Season	**Average price ($)**
Fall	W
Winter	X
Spring	Y
Summer	Z

4 Pivot, aggregate and plot timeseries

1. Import file **TREB_data.xls** as a pandas DataFrame. Create the necessary columns to **pivot** and generate the table below with the value of **Sales** (hint: use the split() function to split the MonthYear column to Month and Year).

2. In **one figure**, plot **5 lines** (one line per year) with the monthly sales versus the month of the year.

3. Calculate the total **Sales** per year and create a bar chart. Do this in **two** different ways: a) group by year the table pre-pivoting, b) summing up the five columns post-pivoting

Sales	Year				
Month	2013	2014	2015	2016	2017
January					
⋮					
December					