EXAM3 PYTHON (90 POINTS)

- This is an examination. It is to be your work and your work alone.
- No exchange of information with another human entity in any form is acceptable.
- Reading Google documents to find out how to solve a problem is not acceptable, including using chatGPT.
- It is ok to use class material, notes, your programs, and any other notes you have written for class.
- You can use the online python documentation pages listed at the end of the Python Notes.

These instructions are general for all the scripts that you will write in python3.

```
Instructions
#class header

#Q1
var1=124
print(var1)

#Q2
var2=3+var1
print(var2)
```

```
Deadline and late penalty
The deadline is 8:00pm with 3 min grace period.
After 8:03pm, the penalty is 2 points for each late minute.
Gradescope Exam3 will close at 8:10m.

IF YOU HAVE TECHNICAL ISSUES DURING THE EXAM LET THE PROFESSOR KNOW
```

Policy for cheating: sharing code on the exam or using chatGPT or using google to find out how to solve a problem, is unacceptable and will earn you a 0 and take you straight to the ethics board. Do not share any study documents either!

We give partial points in case of syntax errors.

We **will subtract -0.5 points for missing or incorrect header and/or Q&A for each script.**

After you download the **Exam3-S2024.zip** from Canvas, unzip the file if necessary, and a directory called **Exam3-S2024** will be created. Change into **Exam3-S2024** and start your work in that directory.

1 (50) The provided data[1] set **Hahn1.csv** contains data on the thermal expansion of copper. The dependent variable is the coefficient of thermal expansion, and the independent variable is the temperature in degrees kelvin.

Data: y = coefficient of thermal expansion
x = temperature, degrees kelvin

The fitting model is the following:

$$Eq1. \qquad y = \frac{b_1 + b_2\, x \;+\; b_3\, x^2 \;+ b_4\, x^3}{1 + b_5\, x \;+\; b_6\, x^2 \;+ b_7\, x^3}$$

In this exercise you will fit the data by using Nonlinear Least Squares Regression.
Use this initial guess for the parameters:

b1	1.0776351733E+00
b2	-1.2269296921E-01
b3	4.0863750610E-03
b4	-1.4262662514E-06
b5	-5.7609940901E-03
b6	2.4053735503E-04
b7	-1.2314450199E-07

In a script called **ex1-s24-copper.py** do the following:

Q0 As a comment include this sentence as Q0 for this script.
I promise not to communicate with another human being in any way about this exam.

Q1 (6) Import NumPy, pandas, Matplotlib, and SciPy to fit the data.

Q2 (6) Create the variables x and y to store the data for the temperature and coefficient of thermal expansion, respectively. Use NumPy **loadtxt()** only one time.

If you do not know how to do this part, for a loss of points, import the variables:

```
from back1 import x,y
```

Q3 (11) Fit the data with the fitting model reported in Eq1. Use the provided initial guess values.

If you do not know how to do this part, for a loss of points, import the variables:

```
from back1 import popt,pcov
```

[1] https://www.itl.nist.gov/div898/strd/nls/data/LINKS/DATA/Hahn1.dat

Q4 (2) Print to screen the fitted parameters.

```
[ 1.07763517e+00 -1.22692969e-01  4.08637506e-03 -1.42626625e-06
 -5.76099409e-03  2.40537355e-04 -1.23144502e-07]
```

Q5 (5) Calculate and print to screen the standard errors of the fitted parameters.

```
[1.70701502e-01 1.20002852e-02 2.25083067e-04 2.75780260e-07
 2.47128896e-04 1.04493708e-05 1.30273297e-08]
```

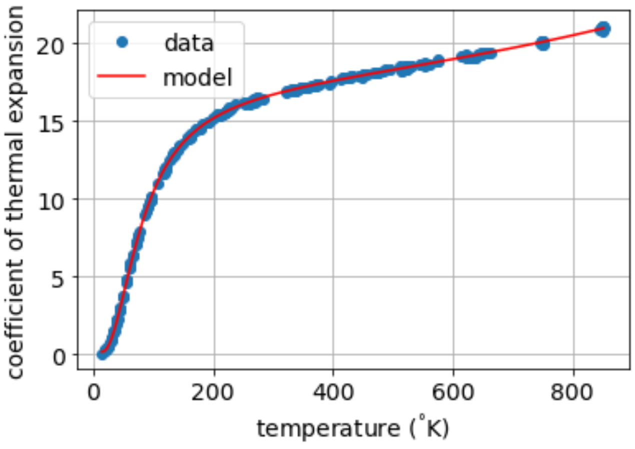
If you do not know how to do this part, for a loss of points, import the variable:
`from back1 import perr`

Q6 (12) Plot the data and the fitted model.
Make the line for the fitted model smooth, by generating 200 points in the x axis, and then use the fitted model to generate the corresponding 200 points in the y axis.
Do not hardcode the beginning and the end of the x values.
If you do not use the Matplotlib Object oriented Interface, you will lose points on this part.

If you do not know how to generate 200 points, for a loss of points, import the variables:
`from back1 import xmodel,ymodel`

Make the figure, and

- set font size to 14
- figure size (6,4)
- label the axes. Notice the degree in Kelvin
- include the legend
- set the grid



Q7 (8) Make this DataFrame, and print it to screen:

```
               estimate     std error
parameter
b1         1.077635e+00  1.707015e-01
b2        -1.226930e-01  1.200029e-02
b3         4.086375e-03  2.250831e-04
b4        -1.426266e-06  2.757803e-07
b5        -5.760994e-03  2.471289e-04
b6         2.405374e-04  1.044937e-05
b7        -1.231445e-07  1.302733e-08
```

The *estimate* column contains the fitted parameters, and the *std error* column the standard error of the parameters.

To make the DataFrame, use existing variables, along with this provided variable:

```
names=['b1','b2','b3','b4','b5','b6','b7']
```

2. (40) The provided dataset[2], **GroundwaterChemistry.csv**, contains results on Groundwater Chemistry near the Kellogg Biological Station Long-Term Ecological Research (KBS LTER), which is funded by the National Science Foundation (NSF). The dataset includes measurements of water chemistry in various surface waters and two water supply wells located in the vicinity of the KBS LTER.

In a script called **ex2-s24-aquatics.py** do the following:

Q1 (3) Import pandas, NumPy and Matplotlib

Q2 (5) Use pandas' **read_csv()** to read in only fields: 3rd, 21st, 22nd, 23rd, and 24th of the provided data set. You should exclude comment lines.
Store the resulting DataFrame in a variable called **df.**

If you do not know how to do this part, for a loss of points, read in this back up file:

```
df=pd.read_csv('GroundwaterChemistry-back2.csv')
```

[2] https://lter.kbs.msu.edu/datatables/390

Q3 (2) Print to screen the first 4 lines of the DataFrame

```
            sampling_point     Ca-IC      Mg-IC     Na-IC      K-IC
0      Pond Lab, Bush Well  85.59397  24.351831  4.199927  0.903582
1  Pond Lab Reservoir well  91.54731  26.826193  4.270796  0.895591
2      Pond Lab, Bush Well  85.30542  24.671526  4.113261  0.916274
3  Pond Lab Reservoir well  83.25407  19.900213  4.121892  0.757301
```

Data are collected in two reservoirs. In this exercise we will focus on the one listed in the *sampling_point* column as *Pond Lab, Bush Well.*

Q4 (7) Make a new DataFrame called **df1**, containing only information on *Pond Lab, Bush Well*, and the chemical elements. For this extract the rows in the *sampling_point* column that are equal to *Pond Lab, Bush Well*, and columns from the *Ca-IC* to the end. Print the Dataframe to screen:

```
        Ca-IC      Mg-IC     Na-IC      K-IC
0    85.59397  24.351831  4.199927  0.903582
2    85.30542  24.671526  4.113261  0.916274
4    76.39045  18.085892  4.008407  0.714832
5    82.02030  22.176386  4.048590  0.777060
8    76.83283  18.764640  3.198460  0.432278
..        ...        ...       ...       ...
128       NaN        NaN       NaN       NaN
130       NaN        NaN       NaN       NaN
131       NaN        NaN       NaN       NaN
133       NaN        NaN       NaN       NaN
134       NaN        NaN       NaN       NaN

[78 rows x 4 columns]
```

If you do not know how to do this part, for a loss of points, import the variable:

```
from back2 import df1
```

This df1 does not contain missing values, but you should assume it does for the next questions.

Q5 (4) Drop rows of df1 containing missing values. After that, the original df1 should not contain missing values.

Q6 (4) Make a new DataFrame called **df1_stats** containing the statistics of **df1**. Use a pandas method.

```
            Ca-IC      Mg-IC      Na-IC       K-IC
count   38.000000  38.000000  38.000000  38.000000
mean    79.207646  24.035881   3.866201   0.829423
std      9.943276   2.427514   0.290500   0.189361
min     35.481194  18.085892   3.198460   0.345811
25%     79.361013  23.393100   3.675494   0.708352
50%     80.887255  23.931480   3.865637   0.844970
75%     81.987347  24.601843   4.006936   0.928914
max     95.311230  34.999615   4.666719   1.226233
```

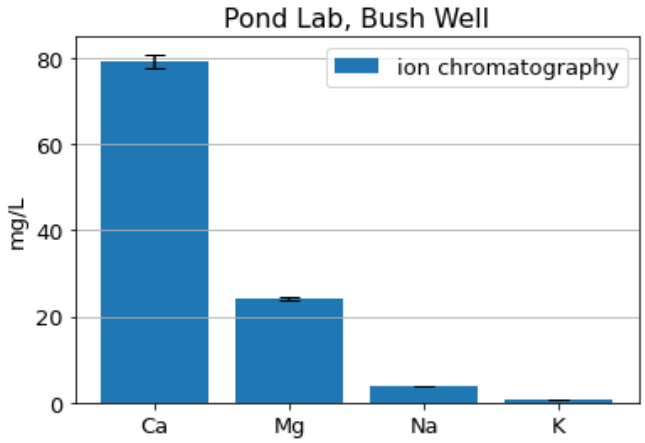
If you do not know how to do this part, for a loss of points, import the variable:

```
from back2 import df1_stats
```

Q7 (15) Use **df1_stats** to make a bar graph with error bars, reporting the average and standard error of each chemical element. Notice the chemical elements are sorted in descending order according to the average.
When you calculate the standard error do not hardcode the number of values.

Note: The x labels do not have the *-IC*, and you should obtain the new labels by using list comprehension.

For the bar graph: label the y axis, make the title, set the font size to 13, include the legend, and the grid, capsize 6.



If you do not know how to make the variables for the bar graph, for a loss of points, import the variables:

```
from back2 import L,y,err
```

```
Submit to Gradescope Exam3:
ex1-s24-copper.py
ex2-s24-aquatics.py
```