EXAM3 PYTHON (90 +3EC 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 in 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 8:30pm with 3 min grace period.
After 8:33pm, the penalty is 2 points for each late minute.
Gradescope Exam3 will close at 8:40m.

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-F2023.zip** from Canvas, unzip the file if necessary, and a directory called **Exam3-F2023** will be created. Change into **Exam3-F2023** and start your work in that directory.

1 (40) The issue of climate change is becoming more and more apparent. In this exercise we focus on Earth's rising surface temperature in USA.
A data set **temperature.csv** was taken from the United States Environmental Protection Agency (EPA) and it contains recoded data on Earth's surface temperature, which includes both land and ocean, in the USA from 1901 to 2021. In the data set the temperature values represent changes with respect to an average, which was set to 0.

Look at the contents of the file, and you will see that the data file contains headerlines, reporting information on the data set, and the data are organized in two fields:

- 1st field the year
- 2nd field temperature in Fahrenheit (F)

One of the tasks of this exercise is to make the figure below, which reports:

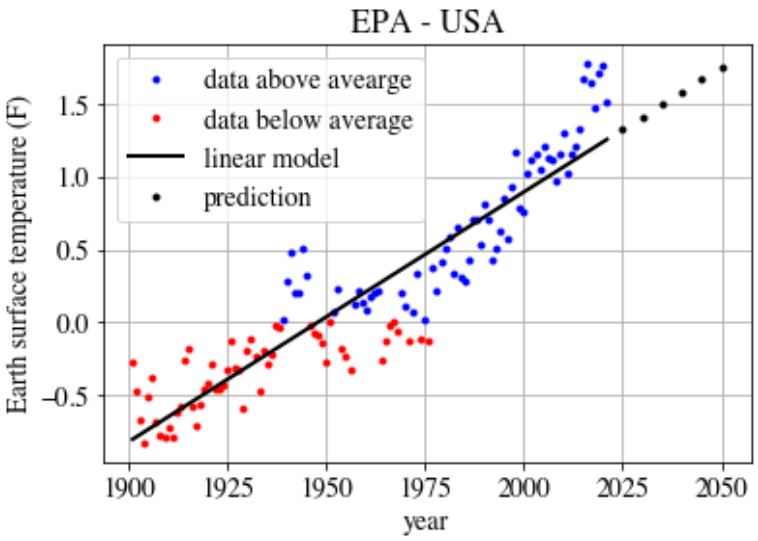
- the data, i.e. the Earth's surface temperature (in Fahrenheit) as a function of years. The red dots report the temperature values below or equal 0 (temperature <=0) and the blue dots the temperature values above 0 (temperature > 0).

- the linear correlation between the temperature and the year depicted with a solid black line. The fitting model used is a polynomial of degree 1.

$$y = \beta_0 x + \beta_1$$

- the prediction depicted with black dots from years 2025 to 2050 (in step of 5 years).

For this exercise use NumPy and Matplotlib.



In a script called **ex1-f23-temperature.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 (1) Import NumPy and Matplotlib.

Q2 (5) Read in data and store the year and temperature values in two separate variables. Use **loadtxt**() to read in data.

If you know how to do this part, for a loss of points, import the variables:
```
from back1 import year,temperature
```

Q3 (6) Generate variables for the data points (red and blue dots). Use Boolean indexing.

If you do not know how to do this part, for a loss of points, import the variables:
```
from back1 import x_pos,y_pos,x_neg,y_neg
```

Q4 (6) Generate variables to make the solid black line reporting the fitted model. You should fit the data and calculate the predicted y values.

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

Q5 (4) Generate variables for the predicted values during years 2025 to 2050 (both included) in step of 5 years.

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

Q6 (10) Make the figure, and
- set font size to 14, font name to Times. Use rcParams.
- use the same colors and markers (all dots) as in the figure.
- use a black solid line for the model.
- label the axes.
- make the title.
- include the legend.
- set the grid.

Q7 (4) Calculate and print the R-squared to screen.

```
The R-squared value is 0.84.
```

If you do not know how to do this part, for a loss of points, import the variable:
```
from back1 import R2
```

Q8 (4) Find and print to screen the last year in which the temperature value was below the average (i.e. below 0). Use indexing. No loops/comprehension, no hardcoding. You should not generate any additional variable but use one variable you previously made.

```
The last year the temperature was below the average was in 1976.
```

If you do not know how to calculate the year, you can hardcode it for a loss of points.

2. (50) The data file **grades.dat** contains information about students' grades. The fields are:

- 1st name of students
- 2nd quiz score
- 3rd exam score
- 4th HW score

Examine the contents of the data file, observing the field separator and the labels for each field. For this exercise, use pandas and Matplotlib.

Make a script called **ex2-f23-grades.py** and in it do the following:

Q1 (1) Import pandas and Matplotlib.

Q2 (3) Use **read_csv** to read in the data from the file **grades.dat** and store the results in a DataFrame called **df**. Print the first 5 lines of **df** to the screen.

The output should be:

```
        name*  qu*iz  ex*am    HW*
0    Susan M.   54.1   83.2   89.9
1  Gregory S.   64.1   51.4  100.0
2   Aleena F.   77.5   93.1   98.0
3  Marylin K.   98.4  100.0   94.0
4  Michael G.   99.6  100.0  100.0
```

If you do not know how to do this part, for a loss of points, import the variable like this:
```
from back2 import df
```

Q3 (2) Display the column labels as a 1D array on the screen.

```
['name*' 'qu*iz' 'ex*am' 'HW*']
```

Q4 (6) Remove the * from the column names of the DataFrame **df**. Use list comprehension and a string method. Print the updated DataFrame **df** to the screen. The output should be:

```
            name  quiz   exam     HW
0       Susan M.  54.1   83.2   89.9
1     Gregory S.  64.1   51.4  100.0
2      Aleena F.  77.5   93.1   98.0
3     Marylin K.  98.4  100.0   94.0
4     Michael G.  99.6  100.0  100.0
5     Phillip M.  58.9   88.8   86.8
6       Costa L.  99.3   87.0   94.0
7      Selena G.  74.3   70.9   83.5
8       Sasha S.  90.1   83.7   92.4
9   Michaella C.  57.7   58.8   69.0
10      Sonia S.  71.7   86.9   85.7
11  Stephanie G.  96.1   51.3   98.0
12      Peter P.  90.2   64.3   70.0
13    Gregory C.  98.9  100.0  100.0
14      Denis M.  80.0   87.5   78.0
```

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

```
from back2 import dfb1
df=dfb1
```

Q5 (4) Use the **describe()** method to output the statistics of **df.** Then, access the output values to extract the *mean* and *std* of the *exam*. Print the result to screen:

```
mean    80.460000
std     16.993814
Name: exam, dtype: float64
```

Q6 (4) Locate the lowest *exam* score and print the corresponding row to the screen as a DataFrame.

```
            name  quiz  exam    HW
11  Stephanie G.  96.1  51.3  98.0
```

Q7 (8) Add a column named *grade* to the DataFrame **df.**
Calculate the grade values by using the following formula:

```
grade=quiz*0.25+exam*0.5+HW*0.25
```

To calculate the grade values, use vectorized operations between columns of **df**.
Print the updated **df** to screen:

```
            name  quiz   exam     HW   grade
0       Susan M.  54.1   83.2   89.9  77.600
1     Gregory S.  64.1   51.4  100.0  66.725
2      Aleena F.  77.5   93.1   98.0  90.425
3     Marylin K.  98.4  100.0   94.0  98.100
4     Michael G.  99.6  100.0  100.0  99.900
5     Phillip M.  58.9   88.8   86.8  80.825
6       Costa L.  99.3   87.0   94.0  91.825
7      Selena G.  74.3   70.9   83.5  74.900
8       Sasha S.  90.1   83.7   92.4  87.475
9   Michaella C.  57.7   58.8   69.0  61.075
10      Sonia S.  71.7   86.9   85.7  82.800
11  Stephanie G.  96.1   51.3   98.0  74.175
12      Peter P.  90.2   64.3   70.0  72.200
13    Gregory C.  98.9  100.0  100.0  99.725
14      Denis M.  80.0   87.5   78.0  83.250
```

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

```
from back2 import dfb2
df=dfb2
```

Q8 (8) Extract the *name* and *grade* of students with *grade* > 90 and store the selected DataFrame in a variable called **df1**.
Then, re-index **df1** so that it becomes the result below. Use the **reset_index()** method. Look at the doc page and figure out how to use it:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.reset_index.html

The extraction and re-index can be done in two lines of code. Print **df1** to screen:

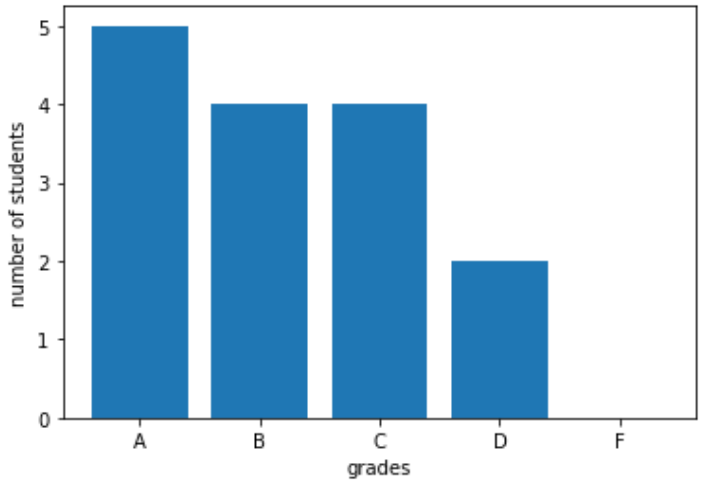
```
          name   grade
0   Aleena F.  90.425
1  Marylin K.  98.100
2  Michael G.  99.900
3    Costa L.  91.825
4  Gregory C.  99.725
```

Q9 (14) Make the bar graph of the grades reported below **by using Matplotlib**.
The x-axis represents the letter grades, and the y-axis the number of students for each letter grade.

```
≥ 90    A
≥ 80    B
≥ 70    C
≥ 60    D
< 60    F
```

To generate the y values, use the DataFrame **df**, Boolean arrays, and a built-in function, without using loop/comprehension.

To generate the x values, you can hardcode them. Label both the x and y axes. No need to set font name etc.
Keep in mind that you need to plot by using Matplotlib, and you need **df** only to generate the variables for the bar graph.



If you know how to calculate the y values, you can use this list, for a loss of points:
`[5, 4, 4, 2, 0]`

EC(3) In a script called **EC3-f23.py**. Use pandas to make a new data file called **sw-all.csv**, which is the concatenation of the following data files:
AZ.dat
CO.dat
NV.dat
CA.dat
NM.dat
UT.dat

Each data file contains information on the temperature anomaly for a southwest state. Look at the contents of all data files (you will recognize these data). Notice that some data files have more than one space between fields, and this was done on purpose, so do not modify the provided data files.

The final data set should be sorted according to temperature values in ascending order, and the field separator should be a comma. Your code should contain one for loop. Generate only the necessary objects. No redundant code will be accepted. You can only hardcode the file names. The contents of the new data file **sw-all.csv** should be:

```
station,state,temp
402,CA,1.138502927
401,CA,1.232179752
501,CO,1.280255682
2601,NV,1.297301136
2602,NV,1.306822486
505,CO,1.338985021
2902,NM,1.341059745
403,CA,1.349018595
2904,NM,1.38277376
4201,UT,1.385726584
2901,NM,1.386311983
504,CO,1.454614325
405,CA,1.457046315
4203,UT,1.461621901
2903,NM,1.480242769
202,AZ,1.480707645
207,AZ,1.503546832
2908,NM,1.513524449
404,CA,1.525912534
4204,UT,1.547443182
4202,UT,1.570402893
503,CO,1.571410124
4205,UT,1.578305785
201,AZ,1.602087638
2905,NM,1.668900654
2907,NM,1.70802772
2604,NV,1.724225207
2906,NM,1.725641357
203,AZ,1.78030303
407,CA,1.781590048
204,AZ,1.806344697
502,CO,1.808518423
2603,NV,1.885235882
205,AZ,1.889656508
4207,UT,1.901579718
206,AZ,1.90334022
406,CA,1.98350551
4206,UT,1.98931646
```

Submit to Gradescope Exam3:
ex1-f23-temperature.py
ex2-f23-grades.py
EC3-f23.py #if you made it
DO NOT SUBMIT ANY ADDITIONAL FILE