EXAM3 - PYTHON (100 POINTS + 3 EC)

- 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.
- It is acceptable to read Google documents; however, no exchanges via the internet are allowed.
- It is permissible to use class materials, notes, your programs, and any other notes you have written for class.
- It is acceptable to use a textbook.
- Sharing your notes with another student is not allowed.
- Any suspected cheating will be reported to the Ethics Board.

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

```
# class header

#Q1
var1=124

#Q2
var2=3+var1
```

```
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 8:40pm

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

Policy for cheating: sharing code on the exam is unacceptable and will earn you a 0 and you and the other person(s) involved will be reported 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-S2023.zip** from Canvas, unzip the file if necessary, and a directory called **Exam3-S2023** will be created. Change into Exam3-S2023 and start your work in that directory.

1. **(35)** The provided data file **video_games.csv** contains information on video games. The first line of the file reports the names of fields.

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

   As a comment write this honor sentence:
   I will not communicate in any way, with any other person about this exam.

Q1 (1) Import pandas

Q2 (2) Read in the dataset **video_games.csv** by using the pandas' **read_csv** function and store data in a DataFrame called df. The first line of the file should label the columns of the DataFrame.

Q3 (2) Print to screen the number of rows and number of columns of df. Use an attribute.

```
There are 16719 rows and 11 columns
```

Q4 (2) Delete possible duplicates from df.

Q5 (2) Have you deleted any duplicates? Answer this question by using an attribute, and print to screen:

```
Yes, I have deleted duplicates, and now there are 16678 rows in the
DataFrame df
```

Q6 (2) Calculate the total number of nulls in each column of df and print the result to screen.

```
Name                  2
Year_of_Release     269
Genre                 2
Publisher            54
NA_Sales              0
EU_Sales              0
JP_Sales              0
Global_Sales          0
Critic_Score       8543
User_Score         9088
Developer          6595
dtype: int64
```

Q7 (2) Delete any row with missing values from df.
After you delete, the DataFrame df should not contain any missing values.

Q8 (2) Print to screen the ndarray of column labels. Use attributes. Print the result to screen.

```
['Name' 'Year_of_Release' 'Genre' 'Publisher' 'NA_Sales' 'EU_Sales'
 'JP_Sales' 'Global_Sales' 'Critic_Score' 'User_Score' 'Developer']
```

Q9 (6) Delete columns NA_Sales, Critic_Score, User_Score and Global_Sales from df by using one line of code. The updated df should not contain these columns.

Then, display to screen the first 2 rows of the updated DataFrame df.

```
             Name  Year_of_Release   Genre Publisher  EU_Sales  JP_Sales  \
0      Wii Sports           2006.0  Sports  Nintendo     28.96      3.77
2  Mario Kart Wii           2008.0  Racing  Nintendo     12.76      3.79

  Developer
0  Nintendo
2  Nintendo
```

Q10 (6) Rename the column labels by using list comprehension.
Remove '_Sales', and substitute 'Year_of_Release' with 'Year'. Use string methods within list comprehension.
Then, print to screen the updated column labels.

```
['Name' 'Year' 'Genre' 'Publisher' 'EU' 'JP' 'Developer']
```

Q11 (8) Extract the names and genres of the games released after 2011 (2011 included), and developed by Nintendo, which had a larger sale in JP with respect to EU. Use Boolean Indexing and print to screen the result.

For this do the following:
Make a Boolean Series satisfying all these conditions:
Year >=2011
JP > EU
Developer is Nintendo

Then use the Boolean Series (Boolean Indexing) to select the Name and Genre of the games satisfying those conditions.

```
                                    Name       Genre
73              Animal Crossing: New Leaf  Simulation
247                             Splatoon     Shooter
424                   Super Mario Maker    Platform
457  Animal Crossing: Happy Home Designer  Simulation
```

2. (35) After a person takes a medicine, the amount of drug left in the person's body decreases over time. When testing a new drug, pharmaceutical companies develop mathematical models to quantify this relationship. Suppose a dose of 1000 mg of a certain drug is absorbed by a person's bloodstream. Blood samples are taken every five hours, and the amount of drug remaining in the body is calculated.

Data from an experiment are stored in the provided file **drugtime.csv**.

The first row contains the hours since the drug was administered, and the second row the corresponding amount of drug in the body (in mg).

Write a script called **ex2-drug.py** and in it do the following:

Q1 (1) Import NumPy and Matplotlib

Q2 (3) The data set **drugtime.csv** contains only numbers, and the field separator is a comma. Read in data **drugtime.csv** and store it in a ndarray called *data*. To read in, use a NumPy function we did in class. Print the 2D array screen.

```
[[   0.    5.   10.   20.   25.   30.]
 [1000.  550. -316.  180.  -85.   31.]]
```

If you cannot read in, import data from back.py and use it.

```
from back import data
```

Q3 (4) Store the hours (1st row) in variable *hour* and the amount of remaining drug (2nd row) in variable *amount*.

If you cannot do this part, import hour and amount from back.py and use them.

```
from back import hour,amount
```

Q4 (5) Some drug *amount* values are negative. Your task is to fix the negative values in the ndarray *amount* and turn them into positive. The ndarray *amount* should contain then only positive numbers. Use Boolean Indexing, and function *np.abs*.

If you cannot do this part, import amountp from back.py and use it.

```
from back import amountp
```

Q5 (8) Fit the data with an exponential model (which is a non-linear model) and print to screen the fitting parameters. Use curve_fit() from scipy.optimize.

The exponential model is:

$$y = a\, e^{(b\, x)}$$

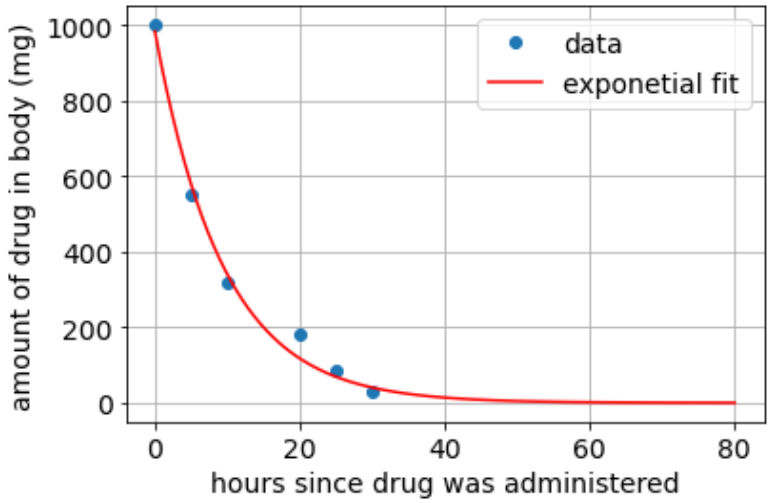
```
[ 9.83372499e+02 -1.06230041e-01]
```

Q6 (9) Plot the data and the fitting model. Plot the data with a marker, and the fitting model with solid line. Pick the colors you like. You can also use default colors.

Plot the fitting model from 0 to 80 hours. To plot the model, generate 100 points on the x axis within this range [0,80] for the hour values, and use the model to calculate the corresponding y predicted values. Use a NumPy function we did in class to generate the 100 points within this range [0,80].

Label the axis, make the grid, and the legend, as in the figure reported below.

Use rcParams to set the font size to 14.



Q7 (5) Estimate the amount of drug left in a body after 5 hours and display the result to screen with the sentence reported below. Format the number with 3 decimal digits. Use the model to predict the amount at 5 hours.

```
The amount of drug after 5 hours is 578.153 mg
```

3. **(30)** Biomedical engineers want to develop an insulin pump for diabetics. To do this, it is important to understand how insulin is cleared from the body after a meal. The concentration of insulin at any time could be described by this equation:

$$C = Co \exp(-\frac{30\ t}{m})$$

where Co is the initial concentration of insulin (in pmol/L), t is the time in minutes, and m is the mass of the person in kilograms.

The provided data set **mypatients.dat** contains names and weights (masses) of some patients.
Here, you will write a script, called **ex3-insulin.py,** that will show how the concentration of insulin after 5 min is different for each person, and depends on the mass of a person.

In **ex3-insulin.py** do the following:

Q1 (3) Import pandas, Matplotlib, and NumPy.

Q2 (2) Read in **mypatients.dat** by using **read_csv()**, and store data in a DataFrame called df. Print df to screen.

```
              name    mass
0   Darby George    63.2
1      Helen Dee    43.5
2  Giovanni Lupa    92.4
3    Cat Donovan    50.1
```

If you cannot do this part, import df from back.py and use it.

```
from back import df
```

Q3 (5) Add a row about the patient "John Smith" with a mass of 45.6 Kg to the DataFrame. Print the updated DataFrame df to screen. Notice the updated DataFrame row indices.

```
              name  mass
0   Darby George  63.2
1      Helen Dee  43.5
2  Giovanni Lupa  92.4
3    Cat Donovan  50.1
4     John Smith  45.6
```

If you cannot do this part, import dfr from back.py and use it.

```
from back import dfr
```

Q4 (8) Use the formula to calculate the concentration C for t=5, and initial concentration Co=85. Use the *np.exp*

Add to the DataFrame df a column reporting the calculated concentration values and label the new column *conc*.
Do not use loops but use vectorization. Print the updated DataFrame to screen.

```
             name  mass       conc
0   Darby George  63.2   7.918751
1      Helen Dee  43.5   2.703035
2  Giovanni Lupa  92.4  16.764686
3    Cat Donovan  50.1   4.257318
4     John Smith  45.6   3.168244
```

If you cannot do this part, import dfc from back.py and use it.

```
from back import dfc
```

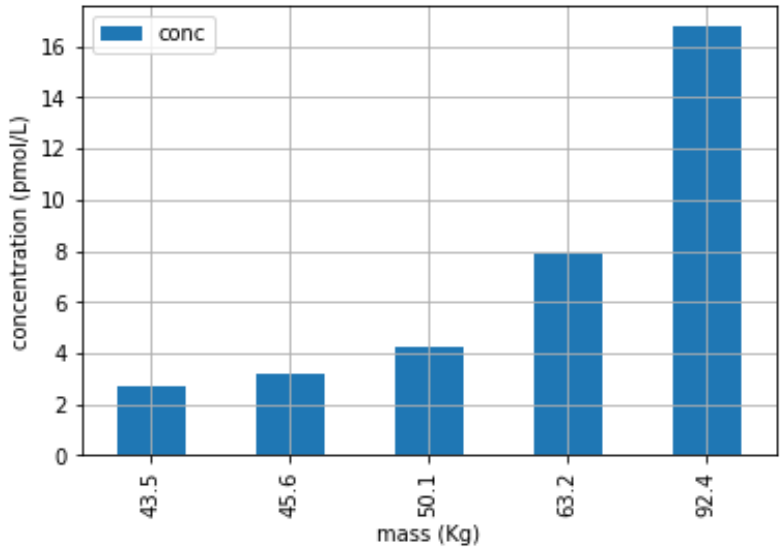
Q5 (4) Sort the DataFrame based on the concentration values, in ascending order. You should modify df. Print df to screen.
Notice the indices of the rows, which are the original indices, so do not re-index.

```
             name  mass       conc
1      Helen Dee  43.5   2.703035
4     John Smith  45.6   3.168244
3    Cat Donovan  50.1   4.257318
0   Darby George  63.2   7.918751
2  Giovanni Lupa  92.4  16.764686
```

If you cannot do this part, import dfs from back.py and use it.

```
from back import dfs
```

Q6 (8) Make this bar graph. Label the x axis and y axis and show the grid.
The x values are the values of the *mass* column, and the y values the values of the *conc* column.



EC (3) In a script called **EC2-S23.py** do the following.
Use a while loop to construct a DataFrame like the one reported below. There are two columns, one labelled *name* and the other labelled *weight*. The values of columns *name* and *weight* must be entered by the user via input functions until the user enters "stop". Thereby, the number of values is not fixed, but it depends on the user entering "stop".
The DataFrame must be constructed within the while loop. No extra loops or compressions should be included. Only one while loop.

This is all or none, and full credits are given to correct code containing only class material.

```
    name  weight
0  Maria    64.4
1    Gio    80.1
2     Do    75.2
```

Submission on Gradescope Exam3 the following:
ex1-games.py
ex2-drug.py
ex3-insulin.py
EC2-S23.py (if you made it)
DO NOT SUBMIT A ZIP FILE.