HW9-PART2 AND PART3 (70 POINTS)

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

```
Instructions for Python3 Homework submission

#class header

#Q1
var1=124

#Q2
var2=3+var1
```

HW9-part2 and HW9-part3 are due Sunday April 7th 11:59pm.

If you submit:

- **HW9-part2 by Wed April 3rd 11:59pm you will get 3 extra credit points.**
- **HW9-part3 by Friday April 5th 11:59pm you will get 3 extra credit points.**

Policy: Work on the HW problems on your own. It is not allowed to share code in any way. The use of chatGPT is not allowed. Using the internet to find out how to solve problems is not allowed.

General Grading rules we will follow:

- We do not give points for instructions but take points off for not following them.
- We take all points off for a question if your code has syntax errors (comment out the code for partial credit).
- We grade the code, not the output of the code. Even if the output is correct, hard-cording, redundant logic, code that has no purpose in the program will not earn full credits.
- We will take points off if you use out-of-class material.
- We will take points off if you do not follow directions when a question specifies to use a certain command or write the code in a certain way.

HW9-part2 (35 points)

1. In this assignment, you will analyze a fictitious dataset called **particles-data.csv** containing measurements from a particle physics experiment, where particle collisions, such as those observed at the Large Hadron Collider (LHC), provide a window into the fundamental properties of matter. The dataset captures the aftermath of these collisions, providing detailed measurements of particle properties such as energy, momentum, charge, and spin. Throughout this assignment, you will explore concepts such as background subtraction to isolate the signals of interest from background noise and conduct a bit of statistical analysis to extract meaningful insights from the dataset.

   Explanation of the dataset fields of particles-data.csv:

   - 1st field Event Number: Unique identifier assigned to each recorded event.
   - 2nd field Energy (GeV): Energy of the detected particle measured in Giga-electronvolts.
   - 3rd field Momentum (GeV/c): Momentum of the particle measured in Giga-electronvolts per speed of light.
   - 4th field Charge (e): Electric charge of the particle measured in elementary charge units.
   - 5th field Spin (ħ): Intrinsic angular momentum or spin of the particle measured in units of the reduced Planck constant.

   Each row represents one event, i.e. one particle detection.

   For this HW part, utilize the NumPy concepts of vectorization, indexing methods, built-in functions, built-in methods, and attributes we covered in class.
   Do not use loops or list comprehensions, unless explicitly specified otherwise.

   In a script **ex9-particles.py** do the following:

Q1 (4) Use the **loadtxt()** function from NumPy to read in the data from the provided file **particle_data.csv** and store it in a NumPy array. Name the array as you like. Ensure that the data is correctly loaded into a 2D array for further analysis.

Q2 (5) Background noise in particle physics experiments refers to unwanted signals recorded by detectors, stemming from various sources such as detector imperfections, environmental radiation, or instrumental artifacts. In the provided dataset, the first 5 rows are designated as background noise. These rows may represent measurements taken before the actual particle collisions or contain data from non-collision events.

   Your task is to generate a clean dataset containing only the data without the background noise.
   To achieve this, perform a background subtraction operation. This involves subtracting the average of the background signal (calculated from the first 5 rows) from the data of interest (rows 6 onwards). The resulting dataset will be free from background noise. Notice that the event number for the first 5 rows is set to 0.

   Here's the step-by-step process:

   Extract the first 5 rows of the dataset and calculate the average for each field (column-wise).

Extract the signal data, which comprises the remaining rows (from the 6th row to the end).
Subtract the average background noise from the signal data element-wise.
You can name your clean data array as desired. Ensure that the clean data remains a 2D array.

For all subsequent questions, utilize the clean dataset generated in the previous step.

Q3 (2) Store each field of the clean dataset in a separate variable. Utilize these variables in the following questions.

Q4 (4) Using NumPy vectorization and Boolean arrays, extract particles that have an energy threshold of 10 GeV and charges greater than zero.

Display the extracted particles on the screen as a 2D array.

```
[[ 9.    11.74  3.7   0.6   0.1 ]
 [10.    27.14  6.3   1.4   0.2 ]]
```

Now, utilize a for loop to iterate over the extracted particles and display each particle individually, i.e. each particle is a 1D array:

```
[ 9.    11.74  3.7   0.6   0.1 ]
[10.    27.14  6.3   1.4   0.2 ]
```

Q5 (4) Find the maximum energy value recorded in the dataset. Print both the maximum value and its corresponding event number. Identifying the particle with the maximum energy provides insights into the most energetic processes occurring in the collision. This understanding aids in comprehending the dynamics of particle interactions at high energies and may reveal rare or exotic phenomena occurring during the collision.

```
Maximum Energy Value is 27.1 Gev:
Event Number where Maximum Energy Occurs: 10
```

Q6 (5) Antimatter particles are identified by their opposite charge compared to their matter counterparts. Extract events involving antimatter particles, i.e. where the charge of the particle is negative (like an antiproton).

Write the extracted events into a file called **antimatter.csv**. The file should be formatted as follows:

```
# Event Number, Energy, Momentum, Charge, Spin
2,-16.4,-0.8,-1.3,0.0
4,7.2,3.3,-0.8,0.2
6,2.4,1.7,-1.4,0.0
8,-10.2,0.4,-1.0,-0.2
11,16.6,4.5,-1.6,-0.1
13,-17.6,-1.6,-1.1,0.0
15,4.3,2.2,-1.8,0.3
18,9.0,3.1,-0.5,0.1
20,-6.6,0.1,-1.3,0.2
```

Look at the doc page of savetxt() to figure out how to obtain this output.

Q7 (2) How many events in the dataset involve antimatter particles?
Print this to screen:

```
There are 9 antimatter events
```

Q8 (2) Calculate the total charge of all particles in the dataset. Print the sum of the charges. Summing the charges provides information about the overall charge distribution and the neutrality of the system under study.

```
Total Charge of All Particles is: -3.9 e
```

e is elementary charge units

Q9 (3) Extract particles with negative momentum and print the event numbers of these particles. Particles with negative momentum represent motion in the opposite direction to the majority of the collision products. Understanding their presence and behavior is crucial for identifying processes such as particle-antiparticle annihilation.

Print to screen:

```
Events with Negative Momentum are:
[ 1  2  3  7 12 13 14 16 19]
```

Use the *np.int64(ndarray)* to convert a ndarray of float to integer type.

Q10 (4) Particle spin is an intrinsic property that influences particle behavior and interactions. Calculate the square root of the positive spin values and print the resulting array.

```
Square Root of Positive Spin Values:
[0.31622777 0.4472136  0.31622777 0.54772256 0.31622777 0.4472136
 0.31622777 0.54772256 0.4472136  0.31622777 0.4472136 ]
```

Upload to Gradescope HW9-part2:
ex9-particles.py

HW9-part3 (35 points)

1. (35) In the provided script **ex9-plotting-timeseries.py**, a variable called **files** has been defined, and it is reported here.

```
files=['Carbon dioxide emission-MTCO2e.txt', 'Land Temperature Anomalies-Celsius.txt',
'Labor Force Statistics-percent.txt', 'Supercomputer power-FLOPS.txt']
```

The variable **files** contains the names of 4 different time series data sets. For each data set the values were collected during different years.

All these data sets are organized as follow:

- 1st field is the year
- 2nd field is a recorded value
- The field separator is a comma.
- The first line is a header line

In this exercise you will write a function called ***timeseries****,* which returns a customized plot designed to plot a time series data set.
You will also use this function within a for loop to make 4 subplots, one subplot for each data set.
From the variable **files**, you will extract the unit and the topic.

Edit the script **ex9-plotting-timeseries.py** and in it do the following:

Q1 (2) Import NumPy and Pyplot from Matplotlib

Q2 (10) Make a function called **timeseries.** This function takes the following 7 parameters:

- ax, one Axes Object
- time, time values
- value, recorded values
- str_format, a string format containing the marker, color and line type
- xlabel, a string for the x label
- ylabel, a string for y label
- str_legend, a string for the legend

The function does the following:

- Uses the plot method to plot the recorded values (value) versus the time values (time). The plot method uses the string format (str_format) and the string for the legend (str_legend).
- Sets the x label by using the string xlabel
- Sets the y label by using the string ylabel.

The function returns the customized Axes Object ax.

Now you will create lists that you will use within a for loop along with the timeseries function to make the 4 subplots reported below:

Q3 (4) Make the following lists out of the list **files** by using for loop or list comprehension.
One list will store the units, and another the topics.
Store them in two separate variables.

```
['MTCO2e', 'Celsius', 'percent', 'FLOPS']

['Carbon dioxide emission', 'Land Temperature Anomalies', 'Labor Force
Statistics', 'Supercomputer power']
```

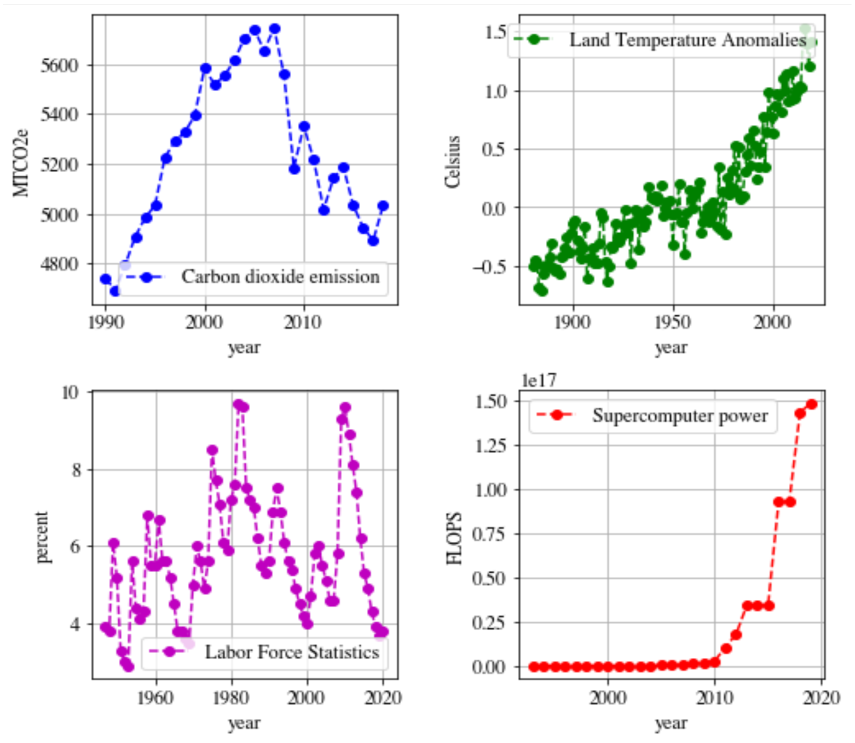
In addition, make this list by just typing:
`formats=['ob--','og--', 'om--','or--']`

Now you will create the subplots.

Q4 (2) Use rcParams to set the font size to 12, the font name to Times (or another font name you like), the width between subplots to 0.4, and the height between subplots to 0.3.

Q5 (4) Create a Figure Object, and a 2x2 matrix of Axes, and set the figure size to (8,8).

Q6 (13) Use one for loop (no nested loops) to loop over the four Axes elements (use enumerate and flatten to turn a 2x2 matrix into a 1D array), and within the loop, use variables files, formats, Lunit, and Ltopic, plus more variables you should generate within the loop.
You should read in data by using loadtxt(), and you should skip the first line of the files. Use the function **timeseries** within the loop.



Upload to Gradescope HW9-part3:
`ex9-plotting-timeseries.py`