Question#1:
Kindly handle the premier league top scorer's information as the following:

- Use Pandas to read the Top_Scorers CSV dataset which contains the following information about top scorers: players names, rank, number of scored goals, and their nationalities
- Use Pandas to read the players_stats JSON dataset which contains the following information about top scorers: players names, height, appearances, wins, losses, goals per match ratio, assists, yellow cards, red cards
- Merge the two datasets
- Visualize a scatter plot about the relation between number of matches played (Appearances) and the number of scored goals

Question#2:
Kindly use Seaborn to do the following:

- Load Titanic dataset
- Visualize a histogram for Age values

Question#3: (30 marks)
Kindly try to prepare the “Telco-customer-churn” dataset

then you can check the following guidelines:

- Use Pandas to read the WA_Fn-UseC_-Telco-Customer-Churn CSV dataset
- set index of the Pandas dataframe by value of CustomerID or drop it
- you may need to convert type of "TotalCharges" column from object to numeric, one of the possible suggestions is to do as the following:
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
    - Take care that the above type conversion will leave some missing values for non-numeric values
    - Why we did this?!, you can try one-hot encoding to know 😊

- replace the missing values in "TotalCharges" by the average of this column
- do one-hot encoding for all the categorical columns, except "Contract" and

"Churn" columns

- analyze "Contract" column, and do ordinal encoding for its values as the following:
  - Replace "Month-to-month" by 1
  - Replace "One year" by 2
  - Replace "Two year" by 3
- convert type of "Contract" column to numeric as we did before, and please don't forget to make sure that there's no missing values left in this column after conversion
- remove duplicates from the dataframe
- do encoding for "Churn" column:
  - Replace "No" by 0
  - Replace "Yes" by 1
- convert type of "Churn" column to numeric as we did before, and please don't forget to make sure that there's no missing values left in this column after conversion
- create a separate dataframe contains the “Churn” values, to act as Targets/Labels while training the machine learning model
  - Take care this new dataframe must has "CustomerID" as it's index also
  - kindly remove "Churn" from the original dataframe
- do normalization for all numerical columns
- save the two prepared dataframes as CSV files

Question#4:

Imagine that you have been asked to design a neural network to classify dogs breeds images,
and while training the model you decided to do an early stopping after 48 epochs to avoid overfitting, but when your manager asked you about the reason you promised to show a plot that will clarify the training and validation losses (Fortunately, you were saving them in each iteration☺). Kindly use the following values saved to plot the losses and show them to your manager.

Please draw both losses in only one subplot, and kindly use a legend.

```
training_losses = [31.935606,
31.679361,
```

```
30.987786,
30.263454,
29.400899,
28.349332,
28.051839,
27.776177,
27.375563,
27.142823,
26.770814,
26.581287,
26.074585,
25.877744,
25.558349,
25.111712,
24.865791,
24.501421,
24.148419,
23.669787,
23.310746,
22.987978,
22.544164,
22.02776,
21.612261,
21.15004,
20.595727,
20.069397,
19.664952,
18.940667,
18.530784,
17.736587,
17.254536,
16.749037,
15.889799,
15.21916,
14.654819,
13.837178,
13.311737,
12.447598,
11.733485,
11.309702,
10.736487,
10.230713,
9.503306,
8.699232,
8.431897,
7.600132]


validation_losses = [5.579518,
5.566535,
5.439262,
5.363996,
5.368029,
5.307929,
5.203864,
5.157347,
```

5.17553,
5.277766,
5.15471,
5.132692,
5.115165,
5.017723,
5.106299,
4.975789,
4.900855,
5.081902,
4.894233,
4.797931,
4.728219,
4.817321,
4.93361,
4.649485,
4.843625,
4.767582,
4.786331,
4.604632,
4.587459,
4.61164,
4.669691,
4.561758,
4.714008,
4.618539,
4.602214,
4.544652,
5.097597,
4.194339,
4.237294,
4.611084,
4.200398,
4.343265,
4.523245,
4.479331,
4.606778,
4.429471,
4.480304,
4.211359]