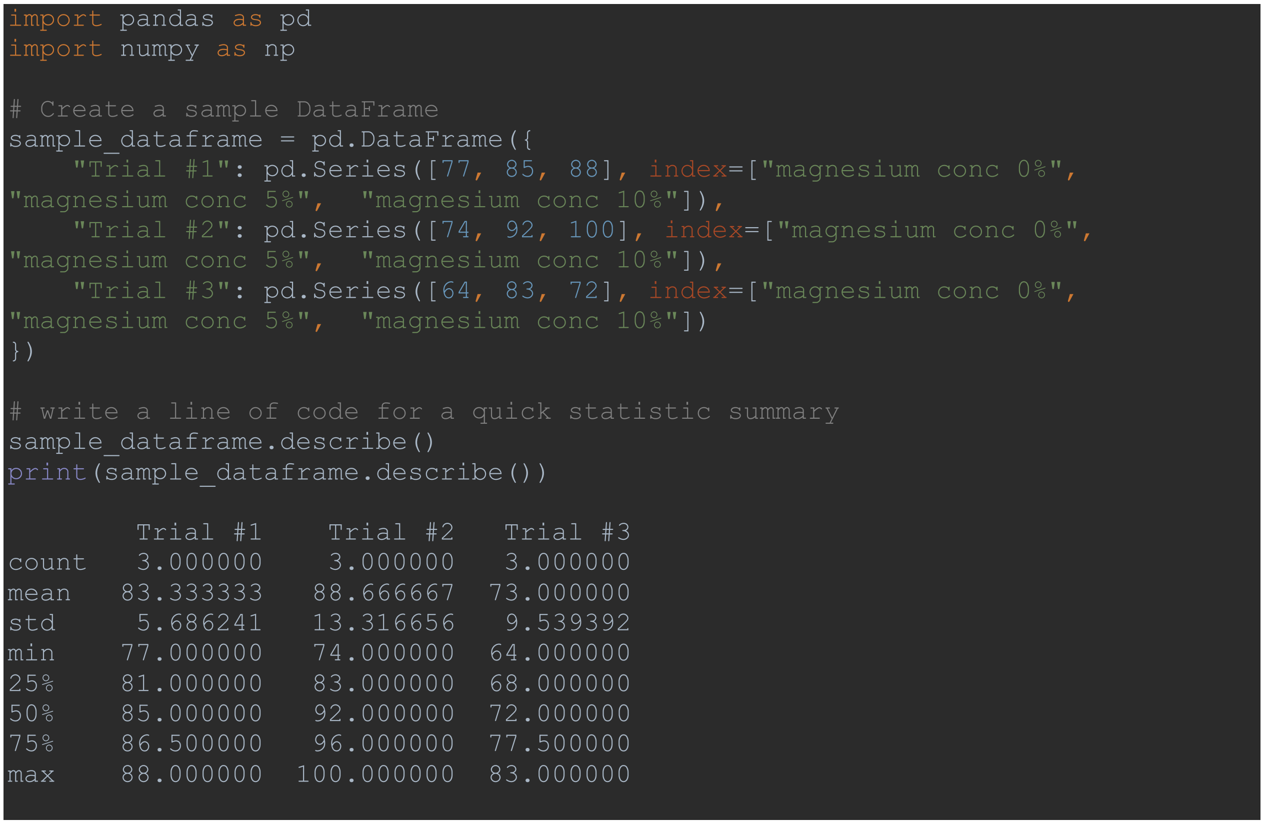
1- What is the difference between a pandas Series and a DataFrame?

Pandas Series	**Pandas DataFrame**
One-dimensional labeled array	Two-dimensional labeled array (with columns and rows)
elements must be of the same data type	elements can have different data types
the size of a Series object cannot be changed	Elements can be dropped or added in an existing DataFrame
Primary building block of a DataFrame (with its columns or rows)	a dictionary of Series objects

2- Create a sample DataFrame and write a line of code for a quick statistic summary of your data in a DataFrame.



```
import pandas as pd
import numpy as np

# Create a sample DataFrame
sample_dataframe = pd.DataFrame({
    "Trial #1": pd.Series([77, 85, 88], index=["magnesium conc 0%",
"magnesium conc 5%",  "magnesium conc 10%"]),
    "Trial #2": pd.Series([74, 92, 100], index=["magnesium conc 0%",
"magnesium conc 5%",  "magnesium conc 10%"]),
    "Trial #3": pd.Series([64, 83, 72], index=["magnesium conc 0%",
"magnesium conc 5%",  "magnesium conc 10%"])
})

# write a line of code for a quick statistic summary
sample_dataframe.describe()
print(sample_dataframe.describe())

        Trial #1    Trial #2   Trial #3
count   3.000000    3.000000   3.000000
mean   83.333333   88.666667  73.000000
std     5.686241   13.316656   9.539392
min    77.000000   74.000000  64.000000
25%    81.000000   83.000000  68.000000
50%    85.000000   92.000000  72.000000
75%    86.500000   96.000000  77.500000
max    88.000000  100.000000  83.000000
```

3- What are the different ways of selecting elements of a DataFrame?

attribute operator .	df.mycolumnname selects single column at a time

index operator[].	df['mycolumnname'] df[1:3] can select rows or a column
loc operator	df.loc[‘rows’, ‘columns’] can select rows and columns by label or name
iloc operator	df.iloc[rows, columns] can select rows and columns by integer position
Boolean operator [] boolean operator loc	df['Myvariablename’] > numericalvalue] df.loc[(df[Mycolumnname'] > numericalvalue)

https://medium.com/epfl-extension-school/selecting-data-from-a-pandas-dataframe-53917dc39953

4- What is sorting for categorical variables in pandas DataFrame based on?

Using sort_values() to perform sorting on the categorial variable of the DF to make them ascending or descending;

```
df.sort_values(‘mycolumnname’)
```

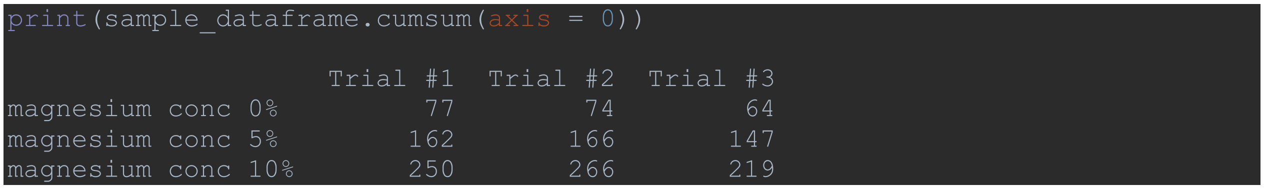
The DF will be sorted based on the values of your column and default to ascending. Where descending is,

```
df.sort_values(‘mycolumnname’, ascending=False)
```

And, the order of your sorting is taken into account, where the first variable will take sorting priority over the second;

```
df.sort_values([‘mycolumnname1’, ‘mycolumnname2’])
```

5- Create a pandas DataFrame containing three columns of randomly generated data. Plot the cumulative sum of each column with labels.



```
print(sample_dataframe.cumsum(axis = 0))

                    Trial #1  Trial #2  Trial #3
magnesium conc 0%         77        74        64
magnesium conc 5%        162       166       147
magnesium conc 10%       250       266       219
```

6- List the age of each gender with the highest weight from the following DataFrame.

```
df = pd.DataFrame({'Name': 'Alex Tom Steve Clarke Sarah'.split(),
   .....:          'Age': [23, 18, 30, 20, 45],
   .....:          'weight': [151, 140, 180, 124, 120],
   .....:          'Gender': ['Male'] * 3 + ['Female'] * 2})
```

```
from IPython.display import display
dict = {'Name': 'Alex Tom Steve Clarke Sarah'.split(),
        'Age': [23, 18, 30, 20, 45], 'weight': [151, 140, 180, 124, 120],
'Gender': ['Male'] * 3 + ['Female'] * 2}
df = pd.DataFrame(dict)
display(df)
#      Name  Age  weight  Gender
# 0    Alex   23     151    Male
# 1     Tom   18     140    Male
# 2   Steve   30     180    Male
# 3  Clarke   20     124  Female
# 4   Sarah   45     120  Female

male_result = df[df["Gender"] == "Male"].sort_values('weight',
ascending=False)["Age"]
female_result = df[df["Gender"] == "Female"].sort_values('weight',
ascending=False)["Age"]
print(f"Male age = {male_result}, Female age = {female_result}")
```

7- Replace the weight values higher than 150 with the mean value of weights in the DataFrame from the previous question.

```
mean_weights = df.weight.mean()
print(f"The mean weight is {mean_weights}")

The mean weight is 143.0

df.loc[df["weight"] > 150, "weight"] = mean_weights
display(df)

     Name  Age  weight  Gender
0    Alex   23     143    Male
1     Tom   18     140    Male
2   Steve   30     143    Male
3  Clarke   20     124  Female
4   Sarah   45     120  Female
```

8- Create a value counts column and reassign back to the following DataFrame.

```
 df = pd.DataFrame({'Animal': 'cat dog dog dog fish'.split(),
   .....:          'weight': [8, 10, 12, 11, 2]})
```

```
df = pd.DataFrame({'Animal': 'cat dog dog dog fish'.split(),'weight': [8, 10,
12, 11, 2]})
df["Animal Count"] = df.groupby(["Animal"]).transform(len)
display(df)
```

```
#   Animal  weight   Animal Count
# 0    cat       8              1
# 1    dog      10              3
# 2    dog      12              3
# 3    dog      11              3
# 4   fish       2              1
```

Reference: https://towardsdatascience.com/when-to-use-pandas-transform-function-df8861aa0dcf