Week 5 - Assignment

Programming for Data Science 2025

Exercises for the topics covered in the fifth lecture.

The exercise will be marked as passed if you get **at least 10/17** points.

Exercises must be handed in via **ILIAS** (Homework assignments). Deliver your submission as a compressed file (zip) containing one .py or .ipynb file with all exercises. The name of both the .zip and the .py/.ipynb file must be *SurnameName* of the two members of the group. Example: Annina Helmy + Markus Anwander = *HelmyAnnina_AnwanderMarkus.zip* .

It's important to use comments to explain your code and show that you're able to take ownership of the exercises and discuss them.

You are not expected to collaborate outside of the group on exercises and submitting other groups' code as your own will result in 0 points.

For question about the lecture content or exam, contact: *annina.helmy@students.unibe.ch* with the subject: *Programming for Data Science 2025 - Lecture XY*. For questions about the excercise/grading of excercises, contact: *thea.waldleben@students.unibe.ch* or *patricia.gribi@students.unibe.ch* with the subject: *Programming for Data Science 2025 - Excercise XY*.

Deadline: 14:00, March 27, 2025.

Exercise 1 - Fitbit dataset 3 points

We will work with three datasets - 'activity.csv', 'calories.csv', and 'last_participant.csv', which contains activity tracker data from https://www.kaggle.com/datasets/arashnic/fitbit

If you are unable to do this exercise, you can load the dataset 'combined_solution.csv' for the next exercise.

1. **Data preparation** (*1 point*)
    - Load the two datasets 'activity.csv' and 'calories.csv'.
    - Use pd.to_datetime to standardize the ActivityDate columns (https://pandas.pydata.org/docs/reference/api/pandas.to_datetime.html)

In [ ]:
```
###
# YOUR CODE HERE
###
```

2. **Merging** (*1 point*)

   - Consider what information is shared between the two datasets and merge them. Keep in mind that the order of rows is not the same in both datasets!
   - Print out the mean "TotalSteps" of the merged DataFrame at this point.

In [ ]:
```
###
# YOUR CODE HERE
###
```

3. **Concatenation** (*1 point*)

   - The data of one additional participant exists in 'last_participant.csv'. Load this dataset and concatenate it with the merged dataset generated above
   - Print out the mean "TotalSteps" again

In [ ]:
```
###
# YOUR CODE HERE
###
```

Exercise 2 - Working with missing data 5 points

In our dataset, some values are missing from the 'TotalSteps' and 'Calories' columns.

We can try to approximate these missing values with the data we got.

You can load the dataset 'combined_solution.csv' if you were unable to complete the previous exercise.

1. **Filling in missing values** (*3 points*)

   - Calculate the mean steps per calory burnt and mean calories burnt per step, by averaging across all observations in the dataset and then computing the ratio. Print out both values.
   - Fill in the null values in the columns 'Calories' and 'TotalSteps' where possible. To fill the values you have to use the factors *"TotalSteps/Calories"* and *"Calories/TotalSteps"* calculated in the previous point, using one of the two information to fill the other.
   - Print out the mean of the columns 'TotalSteps' and 'Calories' before and after filling the missing values.

In [ ]:
```
###
# YOUR CODE HERE
###
```

2. **Dropping missing values** (*2 points*)

   - Print how many null values there are in the 'Calories' and 'TotalSteps' columns, respectively.
   - Drop the rows where **both** 'Calories' and 'TotalSteps' are missing.
   - Print number of rows in the final dataset.

In [ ]:
```
###
# YOUR CODE HERE
###
```

Exercise 3 - Multi-index 7 points

In this exercise you will create and manipulate a multi-index dataframe. First, let's create the dataframe for the exercise:

In [ ]:
```
import pandas as pd

df = pd.DataFrame(
    {
        "idx": [0, 1, 2],
        "A_X": [1.1, 1.1, 1.1],
        "A_Y": [1.2, 1.2, 1.2],
        "B_X": [1.11, 1.11, 1.11],
        "B_Y": [1.22, 1.22, 1.22],
    }
)
```

1. Set the column *idx* as the index of the dataframe. (*1 point*)

In [ ]:
```
###
# YOUR CODE HERE
###
```

2. Create a multi-column stucture. (*3 points*)
   - Set the columns *A, B* on the first level and *X, Y* on the second level, taken from the combinations in the original dataframe.
   - Set the names of the two new levels as "L1" and "L2", respectively.
   - Print the resulting dataframe.

In [ ]:
```
###
# YOUR CODE HERE
```

```
###
```

3. From the previous dataframe, re-create a dataframe with a single column level. (*3 points*)
    - Create a new column from the first level (L1) of the multi-column. At this point your columns should be ['L1', 'X', 'Y'], with name 'L2'. **NB** The DataFrame method *reset_index* is useful for this part.
    - Rename the newly-created column as "letter" and the name of the column level as "L". Use the appropiate pandas methods for this.
    - Print the resulting dataframe.

In [ ]:
```
###
# YOUR CODE HERE
###
```

Exercise 4 - .xs() vs. .loc - on paper 2 points

Consider the following DataFrame:

In [2]:
```
import pandas as pd
data = pd.read_csv('./data/stock_data.csv')
data = data.set_index(["Date", "Ticker"])
data
```

Out[2]:

		Open	High	Low	Close	Volume
Date	**Ticker**					
2018-01-02	**AAPL**	39.986357	40.489241	39.774861	40.479839	102223600
	GOOGL	52.400475	53.543012	52.400475	53.405170	31766000
	MSFT	79.640613	79.807051	79.058082	79.474174	22483800
2018-01-03	**AAPL**	40.543281	41.017967	40.409337	40.472782	118071600
	GOOGL	53.441002	54.544229	53.416123	54.316319	31318000
...	**...**	...	...	...	...	...
2022-12-29	**GOOGL**	86.207842	88.427227	86.197888	88.029129	23333500
	MSFT	231.381200	237.537623	231.381200	236.644104	19770700
2022-12-30	**AAPL**	126.934142	128.456435	125.965402	128.436661	77034200
	GOOGL	86.566129	87.879847	86.158076	87.810181	23986300
	MSFT	233.894827	235.613126	232.372902	235.475662	21938500

3777 rows × 5 columns

1. Select the data for the company 'GOOGL' using the `.xs()` method.
2. Select the data for the company 'AAPL' and the date '2022-12-30' using the `.xs()` method
3. What would the equivalent be using the `.loc` method?
4. What are the main differences between the two methods?
5. When would you use one over the other?
6. How would you pllot the 'Close' price for the company 'AAPL' using the `.xs()` method and matplotlib? Write the code and then check your implementation in your Jupyter Notebook.