Writing

In order to decide on a format in which to store your data, consider the following: what is happening next with your data? If you just took a timestamp and continued working on it with Python, it's best to store the data as a pickle. You can also store a well-named timestamp of your work as a CSV file and simultaneously store a pickle for further work. In general, it is good practice to store reasonable timestamps of your data. For example, once you have finished cleaning the data, store the cleaned data before starting any calculations.

If you want to share your results, a CSV file is usually the format to go, as it is very simple and most programs can handle CSV files. It often makes sense to store the data in the same structure as the original data, as it can then be directly processed further in the same way the original data was intended. Be very strict about naming all your files in a way that you can still deduce their content also long after the project is over. If there are several files, write a README.txt file where you describe the content and the structures of your files. Alternatively, use notebook cells to appropriately describe each step with input and output.

If you want to process and further evaluate a small part of the data, it makes sense to store only parts of the data. It is not always easy to find a balance between consistency and ease of use. If you keep only one big file, it is easy to keep your data consistent, as all the data is handled at once and all changes are applied to all of it. This comes with the downside of carrying along a big chunk of data without much use for it. This is particularly problematic if the data is so big that it does not fit into the working memory of your computer at once. Therefore it is sometimes better to store several smaller files, especially if your data is intended to be fed to a database. Then, you should follow the structure of your data schema (avoiding functional dependencies) or even better directly fill the data to your database (this is covered in the database course CS-A1153).

1. Using Python's Functions

As we did in the first lecture for reading a file, writing to a file implies opening and closing it, or using the with statement. Here is a link to the Python documentation on reading and writing from/to files for more comprehensive descriptions. This time the file must be open with the "w" option, to be able to write to it (and not only read which is what the option "r" implied in the first lecture).

```
file = open("my_file", "w")
file.write("Hello world !")
file.close()
```

Using the with statement for writing:

```
with open("my_file", "w") as f:
    f.write("Hello World !")
```

Remark: think about exception handling when working with a file (see: Errors)

When writing results of intense computations to a file, think about "flushing" regularly so that the information is saved on the disk before the file is closed. Indeed, when writing to a file with Python, the data is actually copied to the buffer and only when the buffer reached a certain size or the file is closed, the data from the buffer is actually written to the file (disk). The flush function copies the string from the buffer to the file. This allows you to save the data onto the disk before closing the file, which can be critical when writing the results of heavy computations to a file.

2. Using Pickles

A Python varable can be saved on the computer in a binary file using the pickle format provided by Python.

Easy examples are given in the Python wiki

```
import pickle
dogs_dict = { 'Ozzy': 3, 'Filou': 8, 'Luna': 5, 'Skippy': 10, 'Barco': 12, 'Balou': 9, 'Laika': 16 }
filename = 'dogs'
outfile = open(filename,'wb')
pickle.dump(dogs_dict,outfile)
outfile.close()
```

Be careful when "unpickling" a PKL file in Python, as the pickle could contain executable code and harm your machine. You should only unpickle data you really trust, and the best practice is to only unpickle files that you actually created yourself.

3. Using Pandas Functions

Pandas provides methods to directly save the results of your work. You can directly create CSV, JSON, XLS or even PKL files from your DataFrames:

- The to_pickle method allows us to save directly the python variable in a repository, so that it can directly be created with the method *read_pickle* in future Python code. Here, the PKL file can only be used via Python code, but it is fast to load since it is already a DataFrame.
- The to_csv method allows us to save our DataFrame as a CSV file. Loading it requires the program to convert a CSV to a DataFrame again, but since it is a CSV file it can also be easily used outside of Python.
- The to_json method allows us to save our DataFrame as a JSON file.
- The to_excel method allows us to save our DataFrame as an XLS file.