Course
CS-E4002
Course materials
Your points
Code Vault

This course has already ended.

« Using Data in Python Course materials Libraries and Packages »
CS-E4002 / Using Data in Python / Data and Input Reading

Data and Input Reading

1. How to get hold of a dataset you want to study:

- Collect the data yourself
- Dataset repositories (Kaggle, phone companies, ...)
- University websites
- Websites for researchers, e.g. society for science
- Governmental agencies, e.g. Nasa
- Look for a dataset via traditional web search (be extra careful)

Wherever you decide to get your dataset from, be aware that the data can be incomplete or dated, and make sure that using the dataset is actually legal.

Before starting to study the dataset, make sure to take notes about where it comes from and what it contains. This should be stated in any work you produce from a dataset.

The example dataset we are using in this course can be found here.

2. How to access the information inside the dataset:

You can open a document with your computer with most text editors and look inside (if the file has the right extension, it gives the type of the data and is recognised by the computer), or you can use Python to print the few first lines of a document and try to find out the type of the document (usually done when the file is missing an extension or opening it fails due to a missing extension):

- The built in open() function (mind the parameter specifying if the file will be only read or will be written into) returns a file on which you can use the function read().

```
file = open('my_file.txt', 'r')
print(file.read())
```

- Once a file has been opened, it is very important to use the close() function to close it. Otherwise, you may lose some of your work if the program exits unexpectedly. You can also run into space troubles if you open too many files at once.

```
file.close()
```

- Instead of using open, read and close, you can use the with statement predefined for files, which closes the file as soon as the block (part of code) is over, thus preventing you from forgetting to close the file (it is a very good habit). Here is a link to a simple example in the documentation.

```
with open('my_results.txt', 'w') as file:
    file.write('value 1')
```

- NB (culture note): Working with files in Python is actually working with streams. The stream page in the Python documentation will give you all of the most common functions you can use on files, and how to use them.

3. Most common data types you will encouter:

- **JSON** (JavaScript Object Notation): Basically the same as a Python dictionary written into a file (different keys and content, the syntax is the same as the one that allows us to define a dictionary in Python).
- **XML** (Extensible Markup Language): Meta markup language (like HTML).
- **CSV** (comma-separated values): Plain text file where the information is organised as in a table. Columns are separated with a comma and a line of the table corresponds to a line of the file. The name tells us everything! CSV is a version of the more general delimiter-separated file concept, which includes formats like tab-separated values. In practice, these formats are the same, just with a different separator character.
- **XLS** (Excel binary file format): Simply a Microsoft Excel file. Other Excel file formats also exist, such as the more modern XLSX and the macro-enabled XLSM.