Code styling rules

Python rules

The coding style is defined in two official guides:

- PEP 8 for the main code text
- PEP 257 for docstring conventions

Recommended IDE is PyCharm, although any text editor may be used (vscode..). It is required that any code that is pushed into shared repository conforms to style guide above. It is recommended to use pylint with autopep8 to format the code before pushing.

Logging in Python

In this project, we will be using **Logging** module, that provides a set of convenience functions for simple logging usage.

To configure the logging, you need to use `basicConfig()` method. You have to provide `format` parameter, that specifies the format of the log message. The format we will use is: `'%(asctime)s - %(levelname)s - %(message)s'`. The following code snipped is an example of basic configuration of logging:

```
import logging
logging.basicConfig(format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)
logger.setLevel(logging.DEBUG)
```

A good convention to use when naming loggers is to use a module-level logger, in each module which uses logging, named as follows: `logger = logging.getLogger(__name__)`

The defined levels, in order of increasing severity, are the following:

Level	When it is used
DEBUG	Detailed information, typically of interest only when diagnosing problems.
INFO	Confirmation that things are working as expected.
WARNING	An indication that something unexpected happened, or indicative of some problem in the near future (e.g. ‘disk space low’). The software is still working as expected.
ERROR	Due to a more serious problem, the software has not been able to perform some function.
CRITICAL	A serious error, indicating that the program itself may be unable to continue running.

`logger.setLevel()` specifies the lowest-severity log message a logger will handle, where debug is the lowest built-in severity level and critical is the highest built-in severity. For example, if the severity level is INFO, the logger will handle only INFO, WARNING, ERROR, and CRITICAL messages and will ignore DEBUG messages.

For more detailed information, see the logging documentation. There is also a good tutorial for logging on this site.

C# rules

We will use ReSharper as automatic corrections program. It can be use for analyzation and set style of consistency rules in the source code written in C#:

- ReSharper - The Visual Studio Extension for .NET Developers (30 day Trial licension, but for students is for free - with ISIC card)

How to install ReSharper into your VisualStudio:

- Download the program from official webpage
- Run the installer file.
- Make sure that the Install option is selected (blue) next to the products you want to install.
- By default, the selected products are installed into all Visual Studio versions on the target machine. If necessary, on the bottom of the installer window, you can deselect some Visual Studio versions. (the selected versions are blue) - we are using **Visual Studio 2017 Professional**
- Then click Install at the bottom of the installer dialog.
- Open the Visual Studio and set the ReSharper Ultimate Shortcuts scheme - Visual Studio
- When Visual Studio is opened, you will see the **ReSharper** menu that appears in Visual Studio menu bar

But now you have turned on the automatic corrections in Visual Studio. Before you are ready to make the commit, you must use these tools in specified order to ensure the C# rules are obeyed.

- For every class/file click on the ReSharper menu -> Edit and use the tools in this order -> Cleanup code, Reformat Code, Apply Code Style and then you can commit the code.

Logging in C#

- NLog from GitHub - download page for free OpenSource program to logging in C#
- NLog - all information about logging program in C#