Notes on good programming style

Marks will be given (or not) for following a good programming style in the assignments. Use spyder or a similar programming environment not notebook (on ADS1).

Structure of the program

The order of the statements should be

1. Import modules first.
2. Define all functions.
3. Then and only then start with the main program (= non-indented lines not being imports or defs).

Python will let you get away with an ugly mix, but that will make the program almost unreadable for humans and might even cause problems and inconsistencies. E.g. you may use functions defined and values assigned to variables in a previous run and might get a crash when running the program the next day.

Imports and functions

- Use dot notation for the modules. Or import only specified functions and constants.
- Every function should have a docstring explaining its purpose and its arguments and give other important usage information as appropriate.

PEP-8

The most important PEP-8 recommendations. Make sure you do not loose marks for not adhering to them.

- One white space after commas, none before.
- No space or one space on both sides before and after binary operators (`+, -, *, /, **`).
- Always one white space before and after assignment `=` and comparison operators ($==, !=, <, <=, >, >=$). The exception: no spaces around keyword argument assignment signs.
- Avoid lines longer than 79 characters. The vertical line in the spyder editor window shows the limit. Use continuation lines instead. (The 79 limit also applies to comments). Continuation lines should be indented.
- Always two empty lines before and after function definitions.
- If you are using spyder you can invoke auto PEP-8 (`Source` → `Autopep8`). This will sort most PEP-8 problems.
- Overlong lines need usually be corrected by hand.

Comments and code clarity.

- Make good use of comments. No student lost marks for too many comments – ever.
- Use meaningful variable names (which can reduce the need for commenting).
- Think about this as "If in one year's time I want to rewrite the program or use parts of it, what information would I need?".
- Use empty lines to give your code an optical structure (see PEP-8).