- Pipetex -

Coding Styleguide

Author
Max Weise
Contact Email: `maxfencing@web.de`

August 25, 2022

Contents

1 Introduction

This document contains a very basic summary and guideline to what is considered „good practice" or „good codingstyle" for this project. This is done for two reasons:

- Future me working on the code surely has forgotten what present me consideres good code and this is a way of reminding future me.
- In the case of multiple contributers I want to help in keeping the code as uniform as possible.

Following, I will give a brief overview of what tools I use and how I define good quality code. The list will surely be incomplete and I'm not entirely sure if I will ever be able to complete it. But additions will be made if needed and requested.

2 Styleguide

2.1 General Guides

As I'm sure the list following in section 2.3 will be incomplete for some time to come, as some situations won't occur or are not forseen at the time of writing. As it would be too much effort to adress every edge case I present the following compromise:

Most situations handling codestyle problems are handled in section 2.3. If for some reason a situation is not handled there, please refer to the Google Python Style Guide. In the rare case that something is not handled *there*, please refer to the official PEP8 Style Guide written by Guido van Rossum. Please note, that neither the style guide provided by Google, not the PEP8 Guide overrule the descitions made in this style guide.

2.2 Tools

Following will list the tools used to help in keeping the code clean and uniform:

2.2.1 Linter

Currently, the **flake8** linter is used. The repository also provides a flake8rc which configures the linter according to the requirements. To run the linter, simply run it on the source directoy using `$ flake8 src/` . I also recomend integrating it into your texteditor or IDE to run it automatically on the edited file or on the press of a button.

As typehints are greatly encouraged in the project, I recomend to use **mypy** when modifying the code. In this case, no configuration is needed and you can simply run `$ mypy src/` on the source directoy to check for any type incompatabilities. This can also be integrated into your texteditor or IDE.

2.3 Style Descitions

<++>