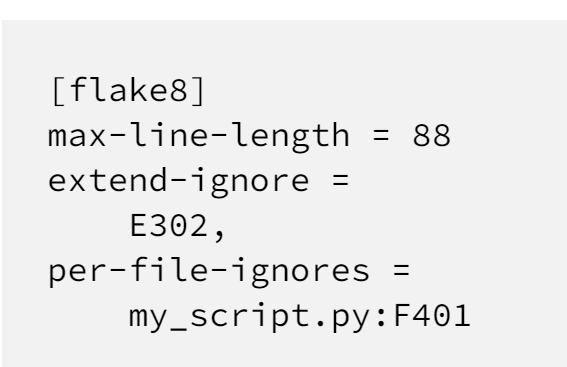
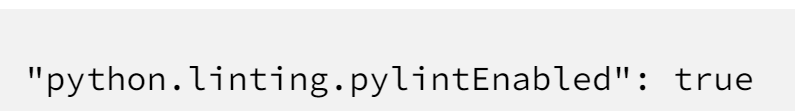
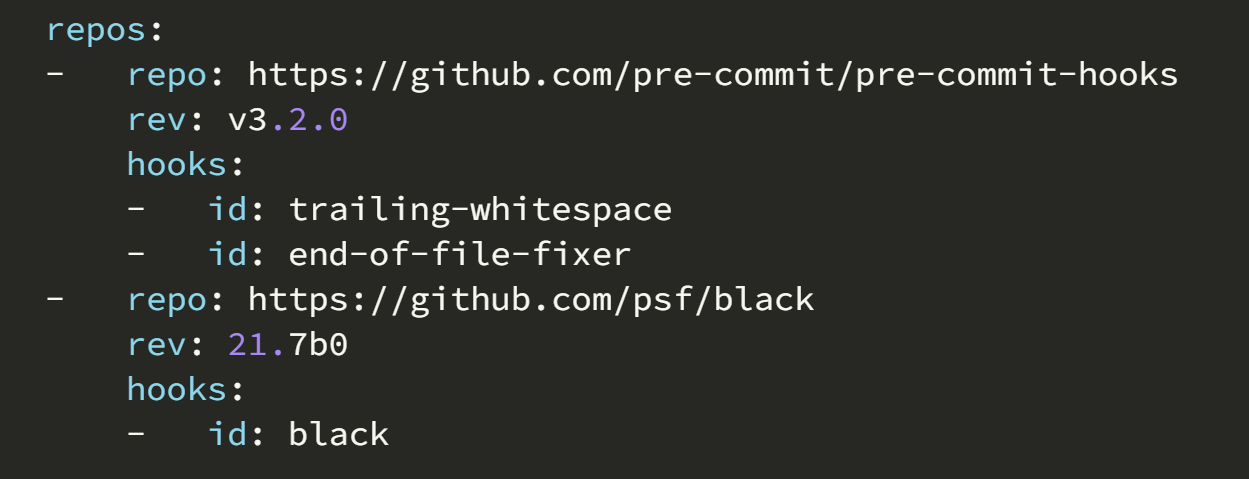
* Linting is a process for identifying bugs and stylistic errors in your code
* Done by analysis tools called ‘linters’
  + flag issues and style violations in your code
* Helps with
  + Readability
  + Debugging
  + Consistency
  + Self-improvement
* Python linters
  + Pylint: looks for errors, enforces a coding standard that is close to PEP8, and even offers simple refactoring suggestions.
  + Flake8: wrapper around PyFlakes, pycodestyle and McCabe; this will check Python source code for errors and violations of some of the PEP8 style conventions.
* Linting in python script
  + pylint
  + Install with: pip install pylint
  + Pylint is used to analyse a Python module
  + run with: pylint my\_script.py
  + Flagged issue format: {path}:{line}:{column}: {msg\_id}: {msg} ({symbol})
  + decide we want to overrule Pylint and ignore a message for a line of code, we can include the comment # pylint: disable=some-message
* Flake8
  + - install with: pip install flake8
    - Run with: flake8 my\_script.py
    - running flake8 will lint all scripts within the current directory and all sub-directories.
    - ignore a particular line of code, you can just add a comment *# noqa* at the end
    - ignore a particular error, you can use, for example, # noqa: F401
    - configure Flake8 so that it will only flag particular errors.
      * adding a setup.cfg file to your working directory
      * Ex config file:
        + for set the maximum line length to be 88;
        + ignore the E302 blank line flags;
        + ignore the F401 flag for my\_script.py only.
        + 
  + Differences /similarities
    - Flake8 is flagging lots of issues related to whitespace and blank lines;
    - Pylint is identifying violations with naming conventions and layout (docstrings, import order, etc);
    - Both linters are pointing out unused imports.
* Linting in editor
  + VSCode: opening the command palette with Ctrl+Shift+P and clicking on Python: Select Linter
  + select which linter we want to us
    - 
* Auto-formatters
  + Follow formatting guidelines automatically
  + Popular autoformatter: Black
    - Installed by: pip install black
    - Run with: black my\_script.py
* Pre-commit hooks
  + Used to implement linters and autoformatters in project
  + pre-commit package manager can be installed with: pip install pre-commit
  + root of our GitHub repo, we then need to create a file called .pre-commit-config.yaml
    - specify the checks we want to run before each commit
  + Ex
    - 
  + After creating .pre-commit-config.yaml file we can then run: pre-commit install
    - whenever the command git commit is run, the pre-commit hooks will automatically be applied