In PyTorch, we enforce lint rules on code in order to help us catch common mistakes and enforce a greater degree of uniformity in our codebase than human reviewers can normally enforce. However, there is nothing more frustrating than to submit a large complicated diff for review, and find out a few hours later that you can't land it because there was some trivial lint problem. **Thus, it is essential to setup your editing environment so that lint rules are applied as you type**, so that you can avoid time wasting situations like this.

NB: Linting on large files (like test_torch.py) can be quite sluggish. See also #18539.

If you work at Facebook; check out this page instead, it details some Facebook peculiarities of linting.

Setting up flake8

We first need to install flake8 with all of the plugins as expected by CI:

```
pip install -r requirements-flake8.txt
```

(These plugins require Python 3. If you use pip3 to install, invoke flake8 with flake8-3.)

Then configure your editor to use it; see below.

VS Code

These instructions were written and tested with standard VS Code, version 1.51.1.

1. Be sure that you have the Python extension for VS Code installed.
2. Set your VS Code workspace to your PyTorch clone directory.
3. Set your Python environment to the one you are using for PyTorch development (usually a `conda` environment) by running the **Python: Select Interpreter** command (which can also be accessed by clicking on the name of the Python version shown in the bar at the bottom of your VS Code window).
4. Enable Flake8 linting by running the **Python: Select Linter** command and choosing "flake8".

After doing the above setup, you should see something similar to this in `.vscode/settings.json` in your local PyTorch clone:

```
{
  "python.linting.flake8Enabled": true,
  "python.linting.enabled": true,
  "python.pythonPath": "/Users/username/miniconda3/envs/pytorch/bin/python"
}
```

If that is the case (rather than, e.g., these settings being set globally), then you should now see PyTorch-specific Flake8 lint results in-editor in VS Code when you edit a Python file in PyTorch, with Python files outside PyTorch being unaffected.

Vim: use ALE

Our official recommendation for language server support in Vim is the **Asynchronous Lint Engine**: https://github.com/w0rp/ale

To install in Vim, first follow the instructions at https://github.com/w0rp/ale#3i-installation-with-vim-package-management

Once you have done so, you must configure which linters ALE will use on PyTorch.

```
let g:ale_linters_explicit = 1
let g:ale_linters = {'python': ['flake8']}
let g:ale_python_flake8_executable = 'flake8-3'
```

ALE populates the vim "quickfix" list, so you can move between errors using :lnext and :lprev. Most vim users will have some keymap setup to conveniently trigger these (e.g., if you use vim-unimpaired, you already have keymaps for these).

Note that this will globally turn on flake8 for all Python files you edit in vim. If you want to work on other Python projects, you will want to make sure these variables only apply when you edit files inside PyTorch. See "Appendix: Per-directory configuration in Vim" for more guidance about how to set this up.

Lint on commit

See https://github.com/pytorch/pytorch/blob/master/CONTRIBUTING.md#pre-commit-tidylinting-hook

Appendix: Per-directory configuration in Vim

There are many ways to go about doing this; here is Edward Yang's personal style using autocmd and functions:

```
let g:ale_linters_explicit = 1
let g:ale_linters = {}

"Create a function per configuration you want"
function! Lint_pytorch()
  let g:ale_linters = {
  \   'python': ['flake8'],
  \}
  let g:ale_python_flake8_executable = 'flake8-3'
```

```
endfunction
"Then call the function via autocmd when you edit a file that matches"
autocmd BufNewFile,BufRead */pytorch*/*.py :call Lint_pytorch()

"Here's another example, for a different project"
function! Lint_ghstack()
  let g:ale_linters = {
  \   'python': ['flake8', 'mypy'],
  \}
  let g:ale_python_flake8_executable = 'flake8-3'
  let g:ale_python_mypy_options = '--strict --config=detailed-mypy.ini'
endfunction
autocmd BufNewFile,BufRead */ghstack*/*.py :call Lint_ghstack()
```

https://stackoverflow.com/questions/456792/vim-apply-settings-on-files-in-directory offers some other ways of doing this. Use whichever style suites you best.