ruff is a package that formats and lints your code

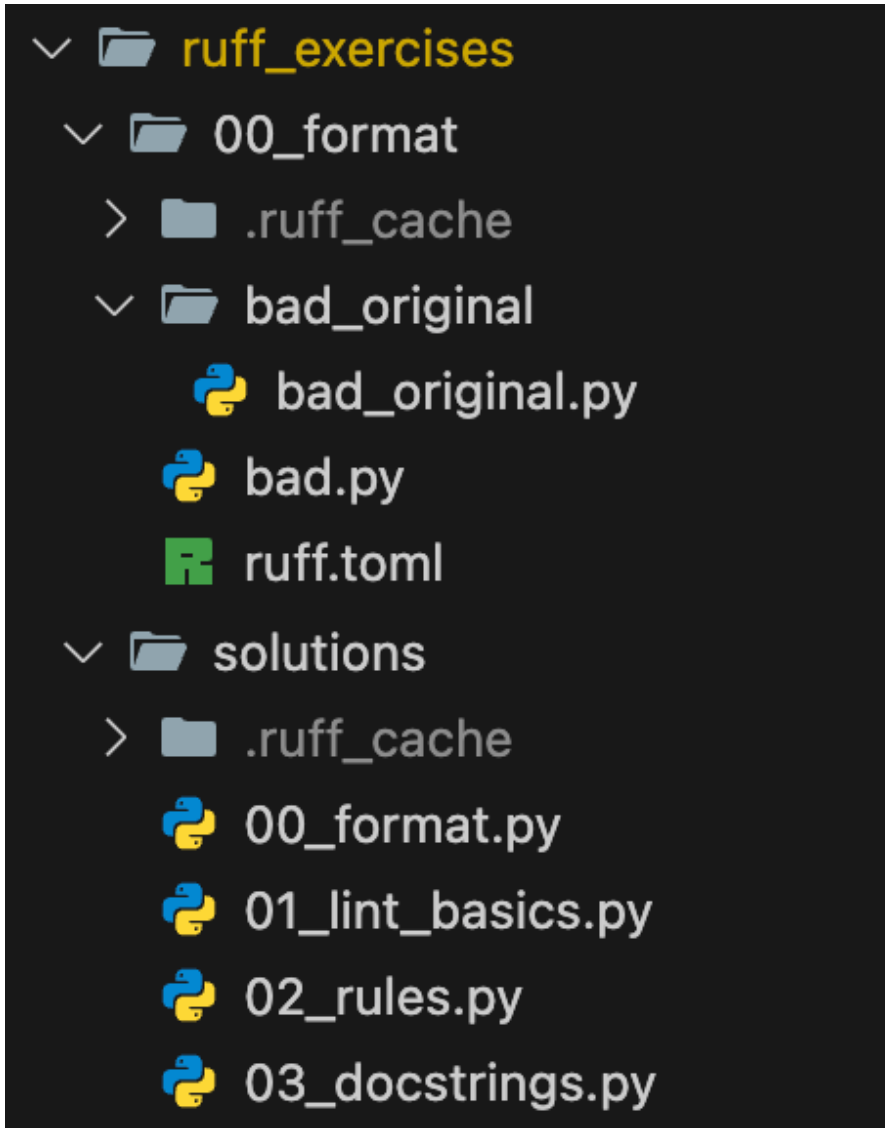
- **Formatting** – reformats code for stylistic consistency
  - e.g., whitespaces, commas, semicolons, line-breaking
  - Saves time when writing code — be as messy as you want!
  - Keeps formatting consistent for other users

- **Linting** – detects stylistic consistency and detects possible errors
  - Poor syntax, errors (e.g., unused imports, undeclared variables)
  - *Keeps formatting consistent for other users*

- Quick to run! (static analysis)
- Can implement an automatic format/lint when using git

Setup

- Each exercise is in its own directory
  - (e.g., `ruff_exercises/00_format/`)
- `bad.py` is the file to be formatted/linted
- `bad_original/bad_original.py` is a copy of `bad.py` to remain unchanged
- The directory `ruff_exercises/solutions` has a properly formatted + linted `bad.py`



00_format

- The purpose of this exercise is to learn how to use `ruff format`

```
$ cd ruff_exercises/00_format
$ ruff format bad.py
```

- Compare `bad.py` with `bad_original.py`
  - What changed?
  - What still needs to be fixed?

01_ruff_basics

- The purpose of this exercise is to learn how to use `ruff check`

```
$ cd ../01_ruff_basics
$ ruff check bad.py
```

After taking a look at the messages, try

```
$ ruff check bad.py --fix
```

- Compare `bad.py` with `bad_original.py`
  - What changed?
  - What still needs to be fixed?

02_rules

- ruff configurations live in ruff.toml — take a peek!
- Comment out **one** of the rules in ignore

```
$ cd ../02_rules
$ vim ruff.toml # rules live here
$ ruff check bad.py
```

- Take a peek at the message and try to make a fix in `bad.py`
- If you don't know what the rule means, check its docs with

```
$ ruff rule B###
```