Линтеры Python

Стилистика кода

Python разрабатывался с идеей что код читают чаще чем его пишут. Поэтому читаемость – один из самых главных принципов написания кода. Для того чтобы наш ленивый мозг мог сконцентрироваться на более важных вещах, используют стандарты стиля кода. Когда разработчики пользуются стандартами, то другим разработчикам проще читать код. Для Python есть практически безальтернативный стандарт стиля – PEP8

Extra: Зачем нужен Code Style?

Линтеры - кто такие и для чего нужны?

Linter - статический анализатор кода, служит для проверки качества кода без его запуска. Обычно линтеры проверяют следующие вещи: Стилистику кода (имена переменных, пробелы, пустые строки) Best practices (использование сырых except, code complexity) Type Checking (расхождение типов данных в аннотациях и в коде) (для языков со статической типизаций таких проблем обычно нет) Линтеры как правило выполняют функцию информирования программиста о проблемах в качестве кода, зачастую линтеры могут также автоматически исправлять распространенные стилистические ошибки.

Какие линтеры существуют для Python

VSCode: крайне слабый встроенный линтер!!!

PyCharm: комбинация из разных линтеров

Standalone linters:

- pep8 - базовая проверка стиля
- pycodestyle
- flake8 - очень распространён
- pylint - популярен для VSCode
- black - существенно строже
- isort - сотировка импортов
- mypy - силён в типизации

Примеры

```
isort .
flake8 --exclude=venv .
pylint --ignore=venv .
deadcode . --exclude=venv
```

flake8/pylance/pylint в VSCode (View -> Problems)

Ruff

Линтер ruff совмещает в себе множество линтеров и имеет поддрежку линтига множества популярных библиотек

```
# Запустить проверку всего кода
ruff check .

# Отсортировать импорты
ruff check --select I --fix .

# Форматировать код
ruff format
```

Пример части pyproject.toml с настроенным ruff

```
[tool.ruff]
target-version = "py311"
line-length = 120
output-format = "concise"
lint.select = ["ALL"]
lint.ignore = [
    "PTH",     # flake8-use-pathlib
    "D",       # pydocstyle
    "FIX",     # flake8-fixme
    "PGH",     # pygrep-hooks (noqa)
    "ERA",     # commented-out-code
    "COM",     # flake8-commas
    "FAST",

    "T201",    # print found
    "E501",    # Line too long
```

```
    "RUF001",  # ambiguous-unicode-character-string
    "TD002",   # missing-todo-author
    "TD003",   # missing-todo-link
    "FBT002",  # boolean-default-value-positional-argument
    "BLE001",  # blind-except
    "S101",    # assert
    "PT009",   # Use a regular `assert` instead of unittest-style
    "RUF100",  # unused-noqa
    "PLR2004", # magic-value-comparison
    "TRY003",  # raise-vanilla-args
    "SIM114",  # if-with-same-arms
    "SIM103",  # needless-bool
    "B904",    # raise-without-from-inside-except
    "RET504",  # unnecessary-assign
    "RET505",  # superfluous-else-return
    "INP001",  # implicit-namespace-package
    "EM102",   # f-string-in-exception
    "S311",    # suspicious-non-cryptographic-random-usage
]
exclude = ["migrations",
        "data",
        "static",
        "data_postgres",
        "logs",
        "drafts"]

[tool.ruff.lint.flake8-annotations]
suppress-none-returning = true

[tool.ruff.lint.flake8-quotes]
inline-quotes = "single"

[tool.ruff.format]
quote-style="single"
exclude = ["Q001"]
```

Revision #7
Created 5 October 2024 18:06:42 by annndruha
Updated 20 March 2026 21:44:36 by annndruha