

Отчет по домашнему заданию 2

Чудова Маргарита

1 План работы

Нужно было выбрать три различных инструмента и настроить их, провести анализ, посмотреть на частые и критичные ошибки.

Я выбрала одну утилиту для стандартизации кода (приведения к виду pep8) — `autoper8`, один линтер — `pylint`, и анализатор кода `bandit`

2 `autoper8`

Эта утилита значительно улучшает внешний вид кода. Я использовала второй уровень строгости и меняла файл на месте (до этого я еще смотрела различия между моей версией и исправленной с помощью `-d`, но там было много и это неудобно для отчета).

Настройки команды: `$ autoper8 -in-place -aggressive -aggressive titanic.py`

Самым частым исправлением был перенос аргументов в функциях, также утилита исправила другие вещи на формат pep8, например поставила пробелы там, где они должны быть. Вряд ли какие-то ошибки можно назвать критичными, главная задача здесь — сделать код более читаемым.

3 `pylint`

Этот линтер хорош тем, что находит как логические, так и стилистические ошибки. Можно попросить его напечатать отчет.

Настройка команды: `$ pylint -ry titanic.py`

Код был оценен на 5.82/10 (без исправлений из `autoper8`). Самыми частыми ошибками оказались `trailing-whitespace` (пробелы не там), `line-too-long` (слишком длинная строка), `invalid-name` (имя не подходит под формат `snake_case`), `wrong-import-position` (надо импортировать модули в правильном порядке), `reimported` (повторный `import`).

Код после `autoper8` был оценен на 4.1/10, при том, что все ошибки типа `trailing-whitespace` и многие другие исчезли. Для меня это остается загадкой.

4 bandit

Bandit делает важную вещь — он исследует код на потенциальные проблемы безопасности. Он достаточно прост в использовании.

Настройки команды: `$ bandit -r titanic.py` (я еще планировала использовать `-lll`, чтобы поварьировать степень серьезности ошибки, но он не нашел никаких ошибок)

Вывод: `No issues identified`, то есть bandit не нашел никаких ошибок.