Linter y formateador de código (en Python)

Linter

Es una herramienta de análisis estático (no ejecuta el programa) que detecta errores, malas prácticas, inconsistencias de estilo y posibles "code smells". Suele emitir advertencias/mensajes y, en algunos casos, puede aplicar autofixes. Ej.: ESLint, StandardJS, Ruff.

Formateador

herramienta que reescribe automáticamente el estilo del código (espacios, saltos de línea, comillas, sangrías, ancho de línea) para hacerlo consistente con una guía de estilos de código (p. ej., PEP (Python Enhancement Proposal) 8). No debería cambiar la semántica. Ej.: Prettier, Ruff Formatter.

Diferencias

Linter: calidad y posibles errores (nombres, imports, complejidad, sombras, etc.).

- **Errores de sintaxis:** Detecta problemas básicos de sintaxis, como la falta de dos puntos y comas, el uso inválido de variables y similares.
- **Infracciones de estilo de codificación:** Comprueba que la sangría, el espaciado y las convenciones de nomenclatura sean correctos.
- **Importaciones, variables y nombres no utilizados:** Marca el código innecesario que puede eliminarse.
- **Complejidad del código:** Advierte sobre funciones demasiado complejas y métodos o clases excesivamente largos.
- **Posibles errores:** Detecta problemas como variables y nombres ocultos, código inaccesible o inactivo, y firmas incorrectas de métodos y funciones.
- **Problemas de gestión de errores:** Identifica excepciones demasiado amplias y bloques de excepción vacíos.
- **Código sin documentar:** Comprueba si faltan cadenas de documentación en módulos, clases, funciones, etc.
- **Mejores prácticas:** Desaconseja las malas prácticas de codificación, como los valores de argumentos predeterminados mutables.
- **Vulnerabilidades de seguridad:** Advierte sobre prácticas de codificación inseguras, como contraseñas codificadas, tokens de API expuestos y el uso de eval() y exec(). Duplicación de código: alerta cuando bloques de código similares aparecen varias veces.

Formateador: apariencia/estilo del código (alineación, comillas, saltos, etc.).

- **Sangría consistente:** Convierte toda la sangría a una sangría estándar de cuatro espacios.
- **Longitud de línea:** Ajusta las líneas que exceden los límites de longitud.
- **Espaciado y relleno:** Agrega o elimina espacios para facilitar la lectura, como alrededor de operadores y después de comas.
- **Consistencia de comillas de cadena:** Convierte todas las cadenas a comillas simples o dobles.

- **Ordenación de importaciones:** Garantiza que las importaciones se ordenen alfabéticamente y se agrupen lógicamente.
- **Uso consistente de líneas en blanco:** Garantiza líneas en blanco entre funciones y clases.