Instituto Tecnológico y de Estudios Superiores de Monterrey

5.2 Ejercicio de programación 2 y análisis estatico

Daniel García Soni
A01796163

Pruebas de software y aseguramiento de la calidad

Fecha:08/02/2025

Contenido

2.7 Diferencia entre Pruebas Dinámicas y Pruebas Estáticas

Las pruebas de software pueden clasificarse en pruebas dinámicas y pruebas estáticas, dependiendo de si requieren la ejecución del código o no.

Característica	Pruebas Estáticas	Pruebas Dinámicas
Ejecución del código	No requiere ejecutar el programa.	Requiere ejecutar el programa.
Momento de aplicación	Antes de la ejecución del software.	Durante la ejecución del software.
Ejemplos	Análisis estático, inspecciones.	Pruebas unitarias, integración, rendimiento.
Objetivo	Identificar errores antes de la ejecución.	Detectar fallos en la lógica del programa.

2.8 Beneficios e Impacto de las Pruebas Estáticas

Las pruebas estáticas aportan los siguientes beneficios al desarrollo de software:

- Detección temprana de errores, reduciendo costos de corrección.
- Mejor mantenimiento y aseguramiento de la calidad del código.
- Reducción del tiempo de depuración en pruebas dinámicas.

2.9 Origen de las Inspecciones en Pruebas Estáticas

Las inspecciones fueron introducidas por Michael Fagan en IBM en 1976 como un método sistemático para encontrar defectos antes de la ejecución del software.

2.10 Diferencias entre Tipos de Revisiones

Tipo de Revisión	Descripción	Ejemplo
Revisión Informal	Se revisa el código de manera casual.	Un desarrollador revisa código sin documentación.
Caminatas Estructuradas	Análisis con checklist en reuniones.	Equipo revisa código con lista de verificación.
Inspecciones	Proceso formal con roles definidos.	Revisión formal con métricas.
Inspecciones Automáticas	Herramientas de análisis estático.	Uso de flake8 y pylint en Python.

2.11 Relación de las Herramientas de Análisis Estático y el Código Fuente

Las herramientas de análisis estático permiten evaluar la calidad del código fuente sin necesidad de ejecutarlo. Detectan errores de sintaxis, problemas de estilo y vulnerabilidades potenciales.

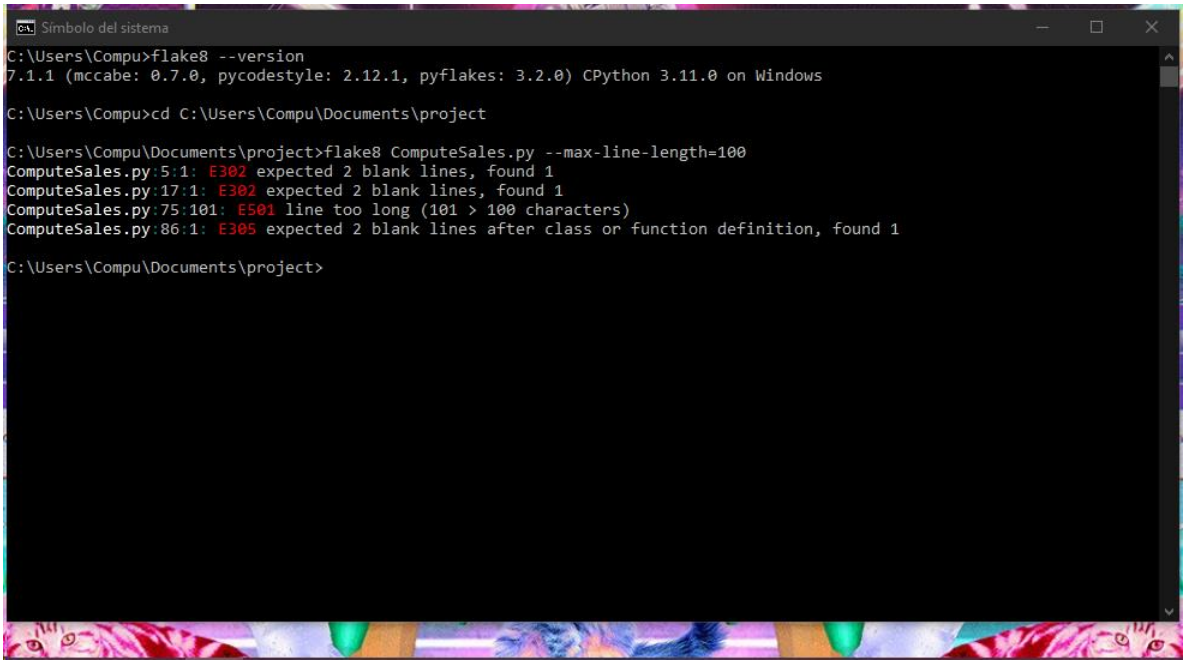
2.12 Experimentación con Herramientas de Análisis Estático

Se utilizaron flake8 y pylint para analizar el archivo compute_sales.py.

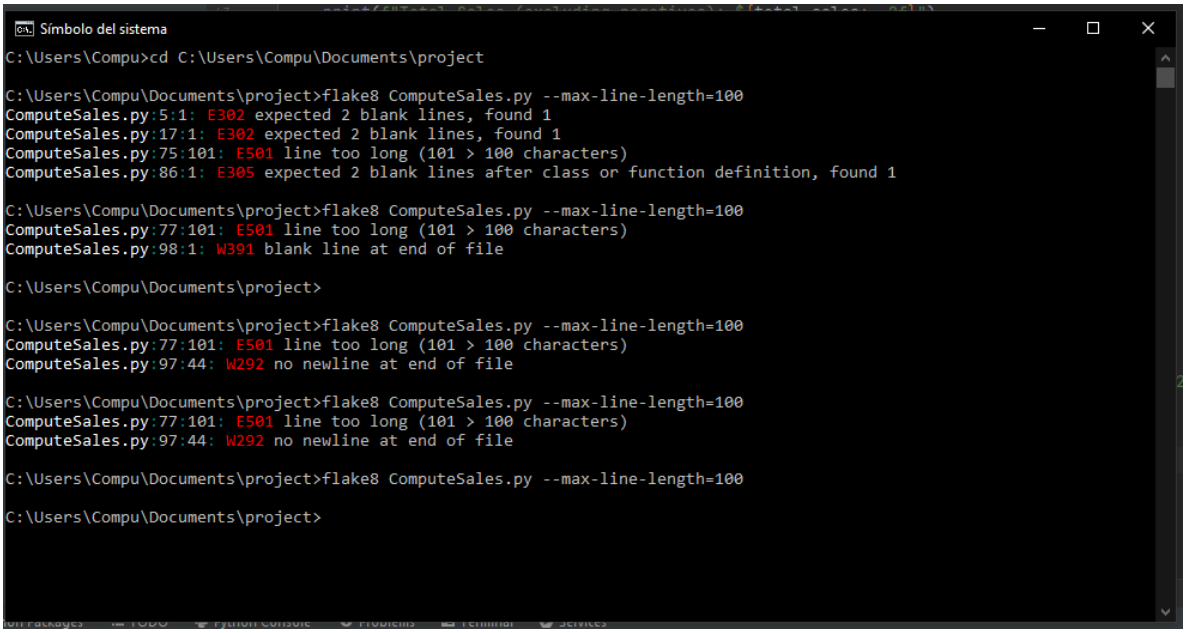
- flake8 compute_sales.py --max-line-length=100
- pylint compute_sales.py

El código fue corregido hasta obtener una puntuación de 10/10 en `pylint`.

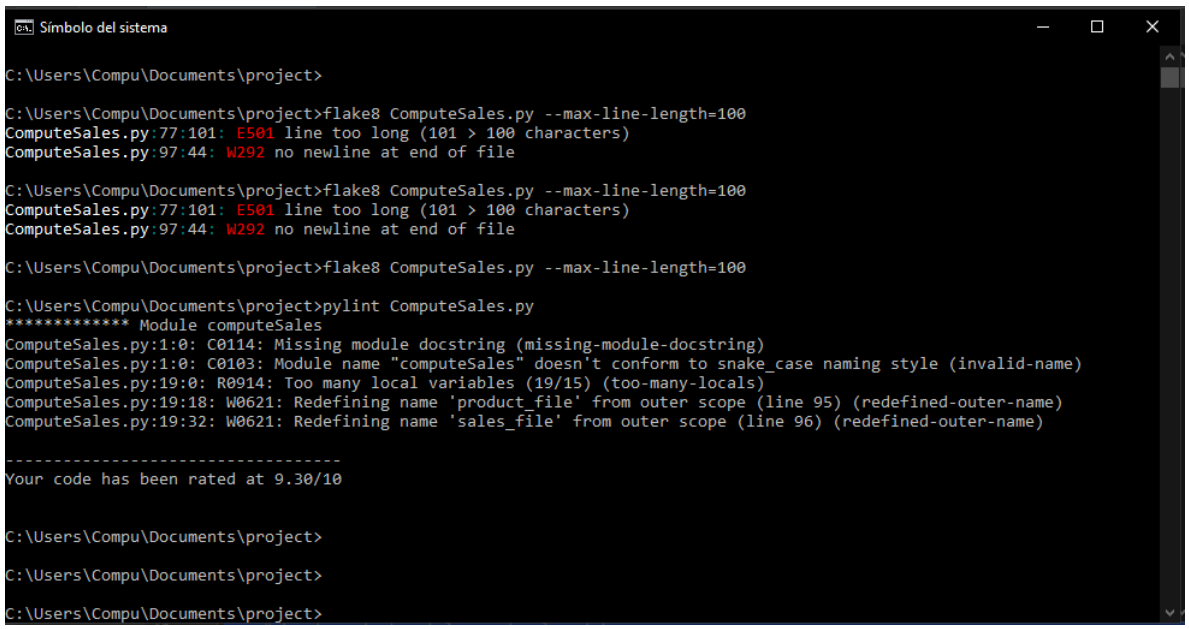
First run flake8:



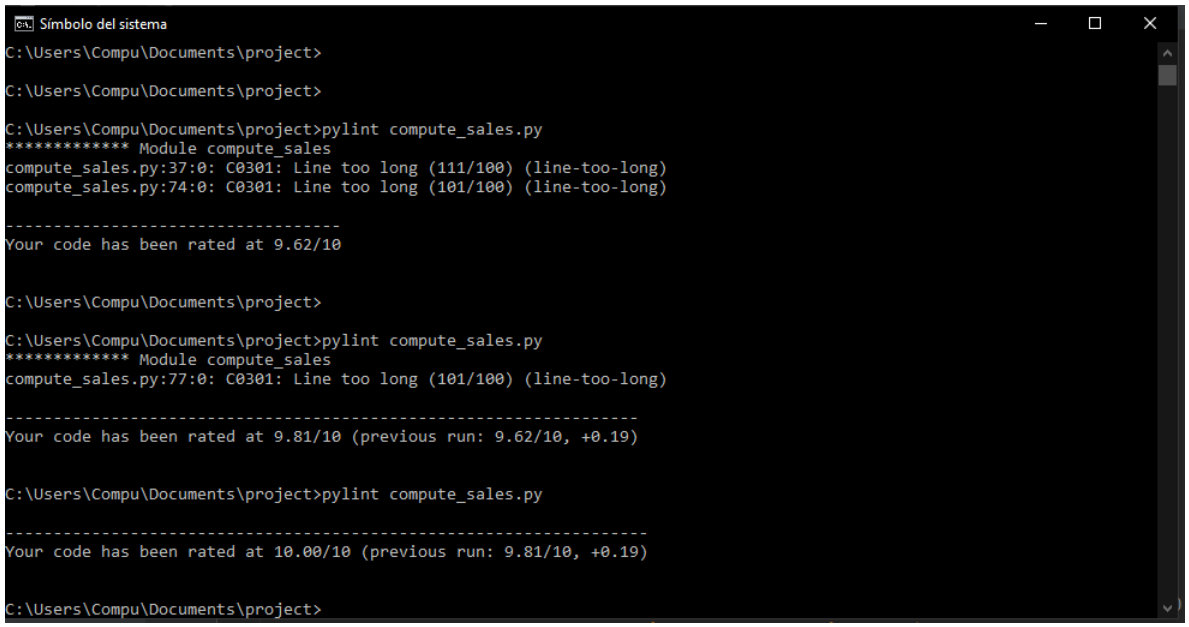
Last flake8 run with no warnings:



First run pylint:



Last pylint run with no warnings:



Referencias

- *Luminous Men, "Python Static Analysis Tools," disponible en: https://luminousmen.com/post/python-static-analysis-tools*
- *Python Software Foundation, "PEP 8 -- Style Guide for Python Code," disponible en: https://peps.python.org/*