

4.2 Ejercicio de programación 1

Daniel García Soni

A01796163

Pruebas de software y aseguramiento de la calidad

Fecha:02/02/2025

Contenido

2.4 Importancia del estilo de la codificación en un sistema	1
2.5 Atributos de un estándar de codificación útil para identificar errores.....	2
2.6 Estándares de codificación reconocidos en la industria y sus implicaciones.....	2
Score first program:	3
Score second program:	3
Score third program:	4
Conclusión	4
Referencias.....	4

2.4 Importancia del estilo de la codificación en un sistema

El estilo de codificación es fundamental para la legibilidad, mantenibilidad y colaboración en el desarrollo de software. Un código bien estructurado permite que los desarrolladores puedan entenderlo fácilmente, incluso si no fueron quienes lo escribieron originalmente. Esto es crucial en proyectos grandes o de larga duración.

Además, un estilo uniforme reduce errores y facilita el mantenimiento, ya que permite detectar problemas con mayor rapidez. Herramientas como *Pylint* en Python ayudan a validar automáticamente el cumplimiento de estos estándares, mejorando la calidad del código (McConnell, 2004).

2.5 Atributos de un estándar de codificación útil para identificar errores

Un buen estándar de codificación debe cumplir con ciertas características clave para minimizar errores y mejorar la eficiencia del desarrollo:

- **Consistencia:** Aplicar reglas uniformes en todo el código.
- **Claridad:** Escribir código simple y directo, evitando redundancias innecesarias.
- **Manejo de errores:** Implementar buenas prácticas para controlar excepciones y evitar fallos inesperados.
- **Compatibilidad con herramientas de análisis:** Usar *Pylint*, *ESLint* u otras herramientas para detectar problemas antes de la ejecución.

Estos atributos garantizan que el código sea más robusto y fácil de depurar, evitando errores costosos en producción (Martin, 2008).

2.6 Estándares de codificación reconocidos en la industria y sus implicaciones

Existen varios estándares ampliamente aceptados en la industria, entre los cuales destacan:

- **PEP 8 (Python):** Define convenciones de estilo para escribir código en Python (Van Rossum, 2001).
- **Google Java Style Guide (Java):** Un conjunto de reglas para mantener consistencia en proyectos de Java.
- **MISRA C (C):** Utilizado en sistemas embebidos y la industria automotriz para garantizar seguridad y confiabilidad.

El uso de estos estándares tiene múltiples beneficios:

1. **Código más confiable:** Se reducen los errores y se mejora la calidad del software.
2. **Facilita la colaboración:** Un estándar común evita inconsistencias cuando varias personas trabajan en el mismo código.
3. **Mantenimiento eficiente:** Un código bien estructurado es más fácil de modificar sin introducir fallos inesperados.

Seguir un estándar de codificación no solo es una buena práctica, sino que **es esencial para el desarrollo de software de calidad y bajo mantenimiento.**

Score first program:

```
Símbolo del sistema

-----
Your code has been rated at 9.74/10 (previous run: 9.61/10, +0.13)

C:\Users\Compu\Documents\pythonProject2\P1>

C:\Users\Compu\Documents\pythonProject2\P1>

C:\Users\Compu\Documents\pythonProject2\P1>python -m pylint compute_statistics.py
***** Module compute_statistics
compute_statistics.py:84:0: C0301: Line too long (105/100) (line-too-long)
compute_statistics.py:123:14: W1309: Using an f-string that does not have any interpolated variables (f-string-without-interpolation)
-----
Your code has been rated at 9.74/10 (previous run: 9.74/10, +0.00)

C:\Users\Compu\Documents\pythonProject2\P1>python -m pylint compute_statistics.py
***** Module compute_statistics
compute_statistics.py:123:14: W1309: Using an f-string that does not have any interpolated variables (f-string-without-interpolation)
-----
Your code has been rated at 9.87/10 (previous run: 9.74/10, +0.13)

C:\Users\Compu\Documents\pythonProject2\P1>python -m pylint compute_statistics.py
-----
Your code has been rated at 10.00/10 (previous run: 9.87/10, +0.13)

C:\Users\Compu\Documents\pythonProject2\P1>
```

Score second program:

```
Símbolo del sistema

wordCount.py:1:0: C0103: Module name "wordCount" doesn't conform to snake_case naming style (invalid-name)
-----
Your code has been rated at 9.86/10 (previous run: 9.47/10, +0.40)

C:\Users\Compu\Documents\pythonProject2\P2>

C:\Users\Compu\Documents\pythonProject2\P2>python -m pylint word_count.py
-----
Your code has been rated at 10.00/10

C:\Users\Compu\Documents\pythonProject2\P2>

C:\Users\Compu\Documents\pythonProject2\P2>cd C:\Users\Compu\Documents\pythonProject2\P3

C:\Users\Compu\Documents\pythonProject2\P3>

C:\Users\Compu\Documents\pythonProject2\P3>

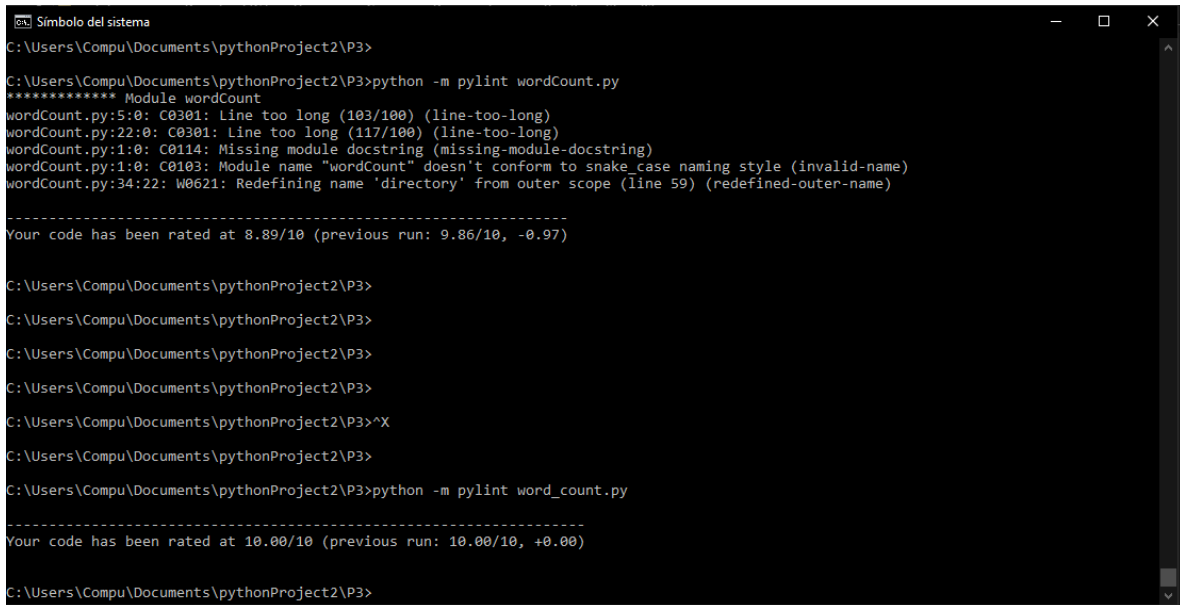
C:\Users\Compu\Documents\pythonProject2\P3>cd C:\Users\Compu\Documents\pythonProject2\P2

C:\Users\Compu\Documents\pythonProject2\P2>

C:\Users\Compu\Documents\pythonProject2\P2>python -m pylint convert_numbers
-----
Your code has been rated at 10.00/10

C:\Users\Compu\Documents\pythonProject2\P2>
```

Score third program:



```
Símbolo del sistema
C:\Users\Compu\Documents\pythonProject2\P3>
C:\Users\Compu\Documents\pythonProject2\P3>python -m pylint wordCount.py
***** Module wordCount
wordCount.py:5:0: C0301: Line too long (103/100) (line-too-long)
wordCount.py:22:0: C0301: Line too long (117/100) (line-too-long)
wordCount.py:1:0: C0114: Missing module docstring (missing-module-docstring)
wordCount.py:1:0: C0103: Module name "wordCount" doesn't conform to snake_case naming style (invalid-name)
wordCount.py:34:22: W0621: Redefining name 'directory' from outer scope (line 59) (redefined-outer-name)

-----
Your code has been rated at 8.89/10 (previous run: 9.86/10, -0.97)

C:\Users\Compu\Documents\pythonProject2\P3>
C:\Users\Compu\Documents\pythonProject2\P3>
C:\Users\Compu\Documents\pythonProject2\P3>
C:\Users\Compu\Documents\pythonProject2\P3>
C:\Users\Compu\Documents\pythonProject2\P3>^X
C:\Users\Compu\Documents\pythonProject2\P3>
C:\Users\Compu\Documents\pythonProject2\P3>python -m pylint word_count.py
-----
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)

C:\Users\Compu\Documents\pythonProject2\P3>
```

Conclusión

En esta actividad, implementé tres programas asegurando que cumplieran con los estándares de codificación PEP 8 y los validé utilizando Pylint para minimizar errores y mejorar la legibilidad del código. Además, realicé pruebas para verificar que los resultados generados fueran correctos en cada caso.

El uso de herramientas como Pylint y la adherencia a estándares de codificación son fundamentales en el desarrollo de software, ya que facilitan el mantenimiento del código, mejoran su claridad y reducen errores en la ejecución. Asimismo, validar los casos de prueba me permitió asegurar que cada programa funciona de manera correcta y genera los resultados esperados.

Finalmente, esta actividad me permitió reforzar mis habilidades en manipulación de archivos, algoritmos básicos y estructuración eficiente del código, además de comprender la importancia de escribir código limpio y bien documentado.

Referencias

- Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.
- McConnell, S. (2004). *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press.
- Van Rossum, G. (2001). *PEP 8 – Style Guide for Python Code*. Python Software Foundation.