



# **Instituto Tecnológico y de Estudios Superiores de Monterrey**



## **Pruebas de Software y Aseguramiento de la Calidad**

### **Análisis de problemas: 4.2 Ejercicio de programación 1**

Julio Adrián Quintana García - A01793661

## Objetivo(s)

- Explicar la importancia del estilo de la codificación de un sistema.
- Reconocer los atributos de un estándar de codificación útil para identificar errores.
- Identificar estándares de codificación reconocidos en la industria y sus implicaciones.

## Instrucciones

En esta actividad vas a generar un repositorio para estos ejercicios de programación en GIT.

Revisa las indicaciones para los programas a implementar:

1. Implementa los programas en este documento usando el lenguaje Python.
2. Sigue el estándar de codificación PEP-8.
3. Verifica la correcta ejecución de sus programas generando pruebas de cada ejercicio usando los recursos indicados. Documenta los resultados.
4. Instala el paquete pyling usando PIP, <https://pypi.org/project/pylint/Links to an external site.>
5. Si tienes duda del uso, revisa el tutorial de PyLint: <https://www.youtube.com/watch?v=fFY5103p5-cLinks to an external site.>
6. Verifica que tus programas no generen errores o problemas usando pylint.

There are 5 kind of message types :

- \* (C) convention, for programming standard violation
  - \* (R) refactor, for bad code smell
  - \* (W) warning, for python specific problems
  - \* (E) error, for probable bugs in the code
  - \* (F) fatal, if an error occurred which prevented pylint from doing
7. Arregla todos los detalles que encontró pylint y verifica que tu programa sigue funcionando correctamente.
  8. Al terminar carga tus programas en tu repositorio personal que generaste. El proyecto deberá de llamarse: Matrícula de estudiante\_Número de actividadA4.2
  9. Sube la liga del repositorio en la tarea de Canvas.
  10. Sube los archivos fuente de la tarea a Canvas.

## Actividad 4.2. Ejercicio de programación 1

Programming Exercise	Description	Practice	Test Cases and Evidence
1. Compute statistics	<p>Req1. The program shall be invoked from a command line. The program shall receive a file as parameter. The file will contain a list of items (presumable numbers).</p> <p>Req 2. The program shall compute all descriptive statistics from a file containing numbers. The results shall be print on a screen and on a file named StatisticsResults.txt. All computation MUST be calculated using the basic algorithms, not functions or libraries. The descriptive statistics are mean, median, mode, standard deviation, and variance.</p> <p>Req 3. The program shall include the mechanism to handle invalid data in the file. Errors should be displayed in the console and the execution must continue.</p> <p>Req 4. The name of the program shall be computeStatistics.py</p> <p>Req 5. The minimum format to invoke the program shall be as follows: python computeStatistics.py fileWithData.txt</p> <p>Req 6. The program shall manage files having from hundreds of items to thousands of items.</p> <p>Req 7. The program should include at the end of the execution the time elapsed for the execution and calculus of the data. This number shall be included in the results file and on the screen.</p> <p>Req 8. Be compliant with PEP8.</p>	<ul style="list-style-type: none"><li>•Control structures</li><li>Console Input output</li><li>• Mathematical computation</li><li>• File management</li><li>• Error handling</li></ul>	Record the execution. Use files included in the assignment

## Proceso de Desarrollo

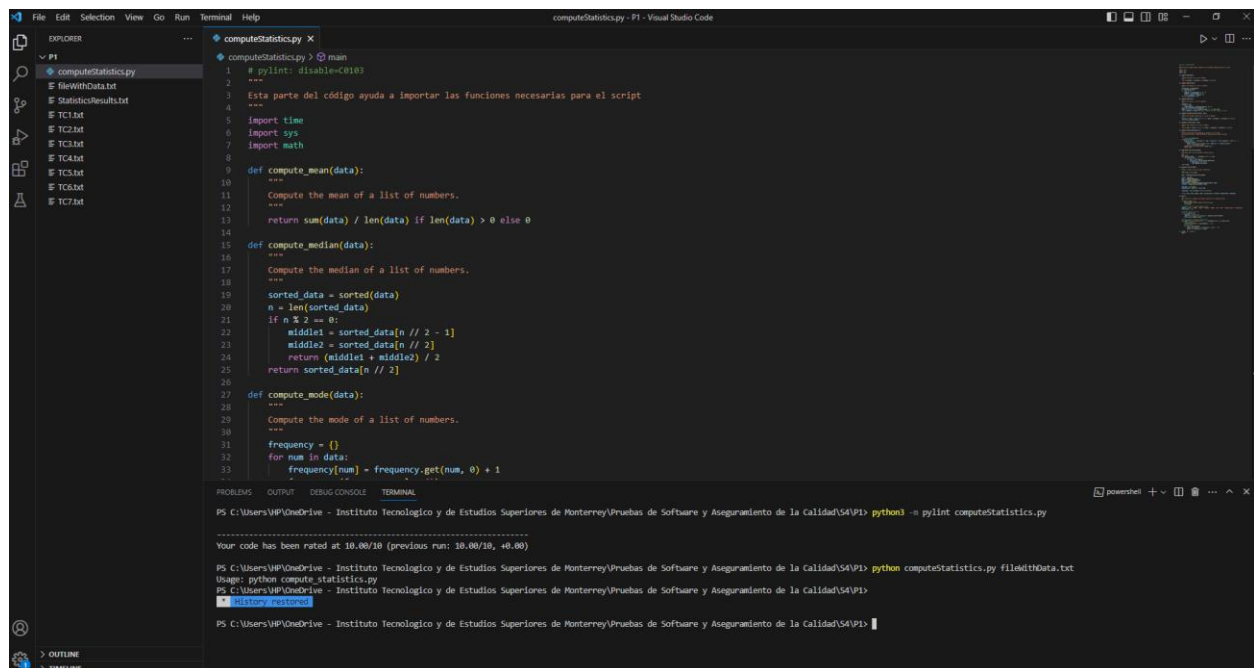
Importancia del estilo de codificación.

El estilo de codificación es importante ya que ayuda a tener legibilidad y mantenibilidad del código. El script sigue convenciones del estilo PEP 8, facilitando su comprensión y colaboración. Sin embargo, podría mejorarse proporcionando comentarios más detallados sobre las funciones y sus propósitos.

El uso de comentarios específicos para deshabilitar advertencias de PyLint (ej. `# pylint: disable=C0103`) es un buen ejemplo. No obstante, el código puede beneficiarse de comentarios adicionales explicando la lógica detrás de los cálculos estadísticos.

## Estándares Reconocidos y sus Implicaciones

El código sigue PEP 8, que es el estándar de estilo de Python. Incorpora buenas prácticas como el uso de espacios en blanco y la longitud de línea máxima. La adopción de estándares reconocidos facilita la colaboración y el mantenimiento del código.



```
1 # pylint: disable=C0103
2 """
3 Esta parte del código ayuda a importar las funciones necesarias para el script
4 """
5 import time
6 import sys
7 import math
8
9 def compute_mean(data):
10     """
11     Compute the mean of a list of numbers.
12     """
13     return sum(data) / len(data) if len(data) > 0 else 0
14
15 def compute_median(data):
16     """
17     Compute the median of a list of numbers.
18     """
19     sorted_data = sorted(data)
20     n = len(sorted_data)
21     if n % 2 == 0:
22         middle1 = sorted_data[n // 2 - 1]
23         middle2 = sorted_data[n // 2]
24         return (middle1 + middle2) / 2
25     return sorted_data[n // 2]
26
27 def compute_mode(data):
28     """
29     Compute the mode of a list of numbers.
30     """
31     frequency = {}
32     for num in data:
33         frequency[num] = frequency.get(num, 0) + 1
34
35 if __name__ == '__main__':
36     fileWithData.txt
37     StatisticsResults.txt
38     TC1.txt
39     TC2.txt
40     TC3.txt
41     TC4.txt
42     TC5.txt
43     TC6.txt
44     TC7.txt
```

Imagen 1.1 – Compute Statistics

## Ejercicio de programación 2 - Converter

Programming Exercise	Description	Practice	Test Cases and Evidence
2. Converter	<p>Req1. The program shall be invoked from a command line. The program shall receive a file as parameter. The file will contain a list of items (presumable numbers).</p> <p>Req 2. The program shall convert the numbers to binary and hexadecimal base. The results shall be print on a screen and on a file named ConversionResults.txt. All computation MUST be calculated using the basic algorithms, not functions or libraries.</p> <p>Req 3. The program shall include the mechanism to handle invalid data in the file. Errors should be displayed in the console and the execution must continue.</p> <p>Req 4. The name of the program shall be convertNumbers.py</p> <p>Req 5. The minimum format to invoke the program shall be as follows: python convertNumbers.py fileWithData.txt</p> <p>Req 6. The program shall manage files having from hundreds of items to thousands of items.</p> <p>Req 7. The program should include at the end of the execution the time elapsed for the execution and calculus of the data. This number shall be included in the results file and on the screen.</p> <p>Req 8. Be compliant with PEP8.</p>	<ul style="list-style-type: none"><li>• Control structures</li><li>• Console Input output</li><li>• Error Handling</li></ul>	Record the execution. Use files included in the assignment

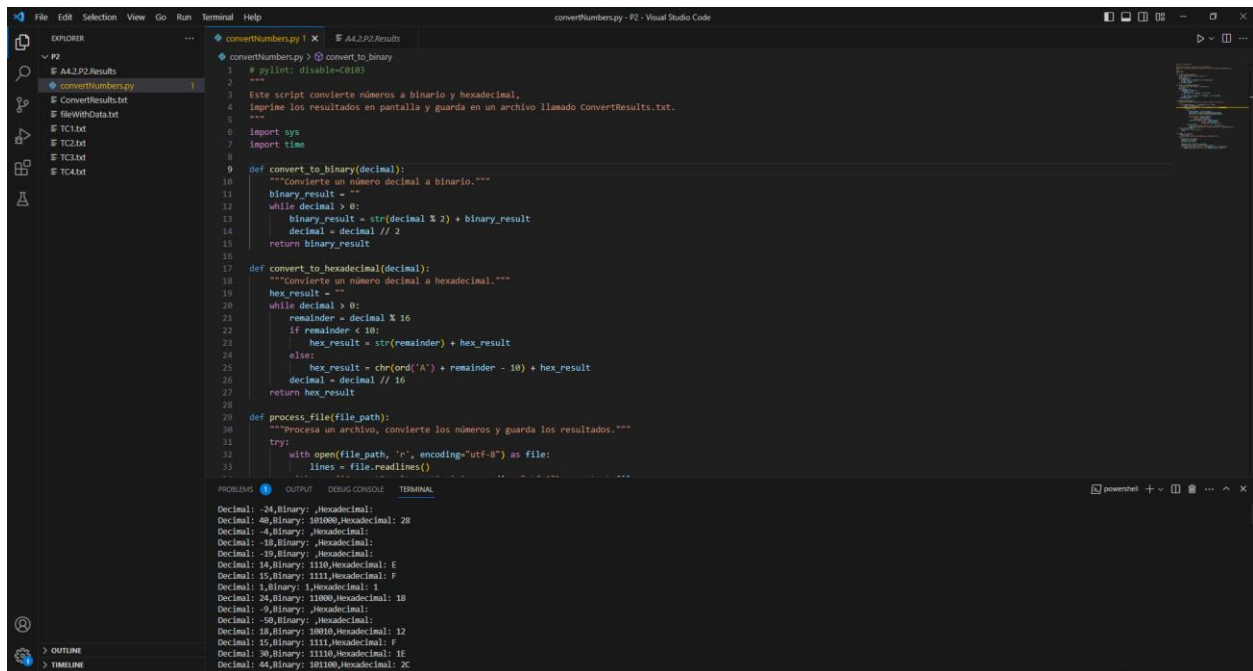
### Importancia del Estilo de Codificación

El script utiliza comentarios para deshabilitar advertencias de PyLint y proporciona una estructura clara. Sin embargo, sería beneficioso agregar más comentarios explicativos dentro de las funciones para mejorar la comprensión del código.

El código maneja errores de manera adecuada, imprimiendo mensajes explicativos para entradas no válidas. Sería útil incluir comentarios sobre cómo los resultados se almacenan en el archivo "ConvertResults.txt".

## Estándares Reconocidos y sus Implicaciones

La inclusión de manejo de errores y la impresión de mensajes claros refleja buenas prácticas de codificación. La extensión del estándar de codificación a través de comentarios más detallados mejoraría aún más la calidad del código.



```
convertNumbers.py
1 # pylint: disable=C0103
2 """
3 Este script convierte números a binario y hexadecimal,
4 imprime los resultados en pantalla y guarda en un archivo llamado ConvertResults.txt.
5 """
6 import sys
7 import time
8
9 def convert_to_binary(decimal):
10     """Convierte un número decimal a binario."""
11     binary_result = ""
12     while decimal > 0:
13         binary_result = str(decimal % 2) + binary_result
14         decimal = decimal // 2
15     return binary_result
16
17 def convert_to_hexadecimal(decimal):
18     """Convierte un número decimal a hexadecimal."""
19     hex_result = ""
20     while decimal > 0:
21         remainder = decimal % 16
22         if remainder < 10:
23             hex_result = str(remainder) + hex_result
24         else:
25             hex_result = chr(ord('A') + remainder - 10) + hex_result
26         decimal = decimal // 16
27     return hex_result
28
29 def process_file(file_path):
30     """Procesa un archivo, convierte los números y guarda los resultados."""
31     try:
32         with open(file_path, 'r', encoding="utf-8") as file:
33             lines = file.readlines()
34
35     except Exception as e:
36         print(f"Error al procesar el archivo: {e}")
37
38
39 if __name__ == "__main__":
40     decimal = -24
41     while decimal < 44:
42         binary = convert_to_binary(decimal)
43         hex_result = convert_to_hexadecimal(decimal)
44         print(f"Decimal: {decimal}, Binary: {binary}, Hexadecimal: {hex_result}")
45         decimal += 1
```

Decimal: -24, Binary: ,Hexadecimal:  
Decimal: 40, Binary: 101000,Hexadecimal: 28  
Decimal: 4, Binary: ,Hexadecimal:  
Decimal: 10, Binary: ,Hexadecimal:  
Decimal: 19, Binary: ,Hexadecimal:  
Decimal: 14, Binary: 1110,Hexadecimal: 6  
Decimal: 15, Binary: 1111,Hexadecimal: F  
Decimal: 1, Binary: 1,Hexadecimal: 1  
Decimal: 24, Binary: 11000,Hexadecimal: 18  
Decimal: 9, Binary: ,Hexadecimal:  
Decimal: 50, Binary: ,Hexadecimal:  
Decimal: 18, Binary: 10010,Hexadecimal: 12  
Decimal: 15, Binary: 1111,Hexadecimal: F  
Decimal: 30, Binary: 11110,Hexadecimal: 1E  
Decimal: 44, Binary: 101100,Hexadecimal: 2C

Imagen 2.1 – Converter

### Ejercicio de programación 3 – Count Words

Programming Exercise	Description	Practice	Test Cases and Evidence
3. Count Words	<p>Req1. The program shall be invoked from a command line. The program shall receive a file as parameter. The file will contain a list of items (presumable numbers).</p> <p>Req 2. The program shall identify all distinct words and the frequency of them (how many times the word “X” appears in the file). The results shall be print on a screen and on a file named WordCountResults.txt.</p> <p><b>All computation MUST be calculated using the basic algorithms, not functions or libraries</b></p> <p>Req 3. The program shall include the mechanism to handle invalid data in the file. Errors should be displayed in the console and the execution must continue.</p> <p>Req 4. The name of the program shall be wordCount.py</p> <p>Req 5. The minimum format to invoke the program shall be as follows: python wordCount.py fileWithData.txt</p> <p>Req 6. The program shall manage files having from hundreds of items to thousands of items.</p> <p>Req 7. The program should include at the end of the execution the time elapsed for the execution and calculus of the data. This number shall be included in the results file and on the screen.</p> <p>Req 8. Be compliant with PEP8.</p>	<ul style="list-style-type: none"><li>• Control structures</li><li>• Console Input output</li><li>• Error Handling</li><li>• String manipulation</li></ul>	Record the execution. Use files included in the assignment

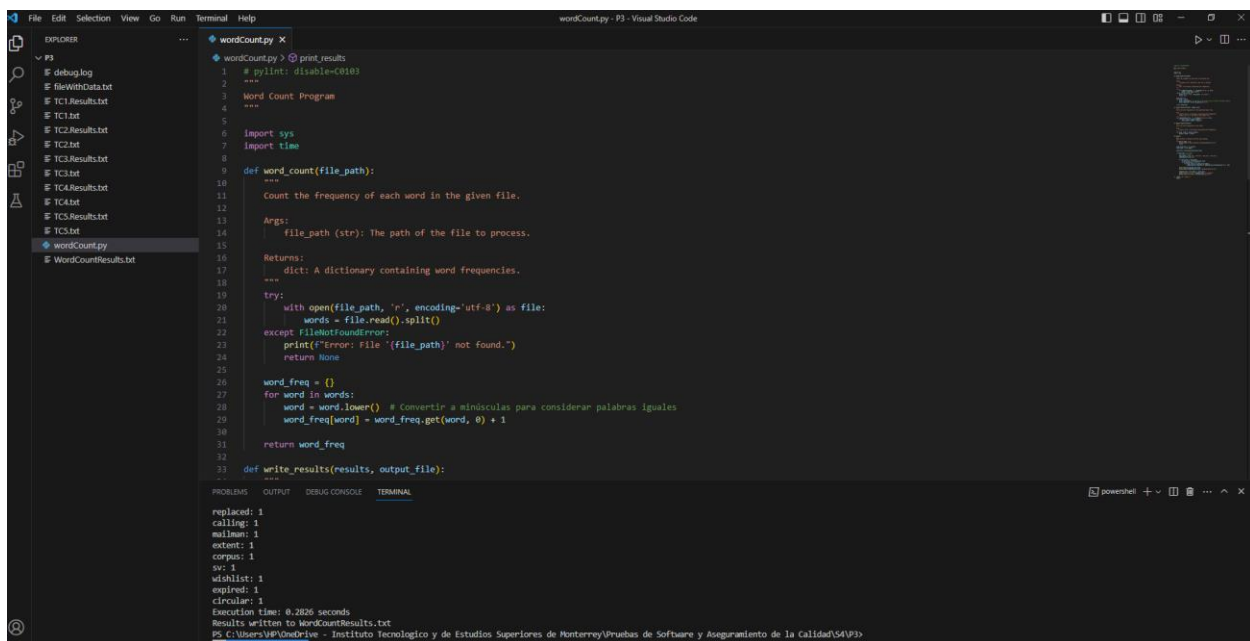
#### Importancia del Estilo de Codificación

El script sigue las convenciones de estilo PEP 8, con nombres de funciones descriptivos. Sin embargo, la ausencia de comentarios podría dificultar la comprensión de la lógica en ciertas partes del código.

El código maneja la lectura de archivos y el procesamiento de palabras eficientemente. Sin embargo, algunos comentarios podrían ser añadidos para explicar la lógica detrás de ciertas decisiones de diseño.

## Estándares Reconocidos y sus Implicaciones

El script sigue el estándar PEP 8 y utiliza funciones descriptivas, contribuyendo a la claridad del código. La inclusión de comentarios explicativos mejoraría aún más la legibilidad y comprensión del código.



```
wordCount.py > @ print_results
1 # pylint: disable=C0103
2 """
3 Word Count Program
4 """
5
6 import sys
7 import time
8
9 def word_count(file_path):
10     """
11     Count the frequency of each word in the given file.
12
13     Args:
14         file_path (str): The path of the file to process.
15
16     Returns:
17         dict: A dictionary containing word frequencies.
18     """
19     try:
20         with open(file_path, 'r', encoding='utf-8') as file:
21             words = file.read().split()
22     except FileNotFoundError:
23         print(f'Error: file {file_path} not found.')
24         return None
25
26     word_freq = {}
27     for word in words:
28         word = word.lower() # Convertir a minúsculas para considerar palabras iguales
29         word_freq[word] = word_freq.get(word, 0) + 1
30
31     return word_freq
32
33 def write_results(results, output_file):
34     """
35     Write the word frequency results to a file.
36
37     Args:
38         results (dict): The word frequency dictionary.
39         output_file (str): The path of the output file.
40     """
41     with open(output_file, 'w', encoding='utf-8') as file:
42         for word, freq in results.items():
43             file.write(f'{word} {freq}\n')
44
45 if __name__ == '__main__':
46     start_time = time.time()
47     file_path = sys.argv[1]
48     results = word_count(file_path)
49     if results:
50         output_file = 'WordCountResults.txt'
51         write_results(results, output_file)
52     end_time = time.time()
53     execution_time = end_time - start_time
54     print(f'Execution time: {execution_time} seconds')
55     print(f'Results written to {output_file}')
56
57 # C:\Users\UP\OneDrive - Instituto Tecnológico y de Estudios Superiores de Monterrey\Pruebas de Software y Aseguramiento de la Calidad\SAVP\3
58 # 10/10/2023 10:10:10 AM
```

replaced: 1  
calling: 1  
mailman: 1  
estent: 1  
corpus: 1  
src: 1  
wishlist: 1  
expired: 1  
circular: 1  
Execution time: 0.2826 seconds  
Results written to WordCountResults.txt  
C:\Users\UP\OneDrive - Instituto Tecnológico y de Estudios Superiores de Monterrey\Pruebas de Software y Aseguramiento de la Calidad\SAVP\3

Imagen 3.1 – Word Count

## Bibliografía:

- PEP 0 – Index of Python Enhancement Proposals (PEPS) | Peps.python.org. (s. f.). <https://peps.python.org/>
- The Python tutorial. (s. f.). Python documentation. <https://docs.python.org/3/tutorial/index.html>
- Real Python. (2017, 13 junio). Pylint Tutorial – How to write clean Python [Vídeo]. YouTube. <https://www.youtube.com/watch?v=fFY5103p5-c>