Materia: **Pruebas de software y aseguramiento de la calidad**

Maestro Titular: **Dr. Gerardo Padilla Zárate**

Maestro Asistente: **Yetnalezi Quintas Ruiz**

Matricula: **A0173101**

Alumno: **Guillermo Alfonso Muñiz Hermosillo**

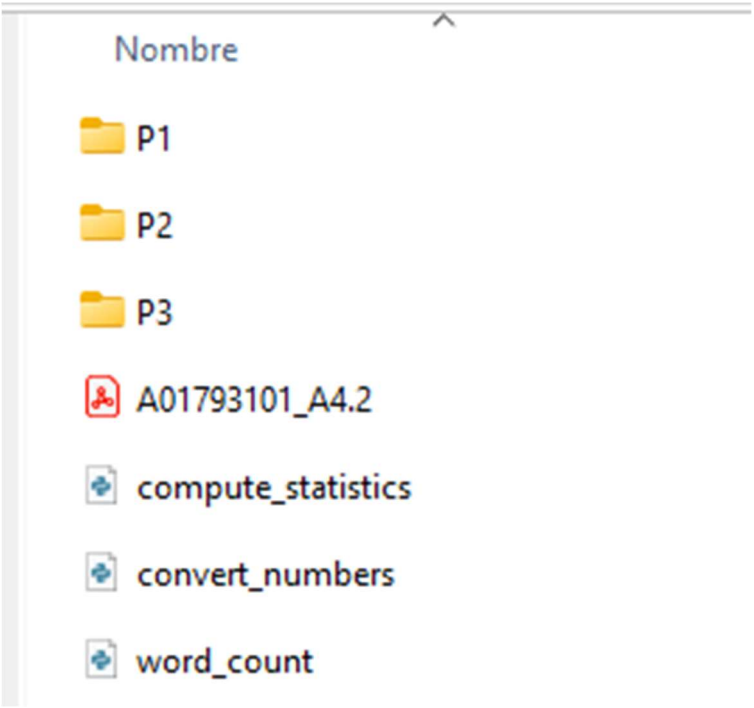
E-mail: A01793101@tec.mx

4.2 Ejercicio de programación 1

Repositorio de Github: https://github.com/A01793101-GMuniz/A01793101_PruebasDeSoftware/tree/main/A01793101_A4.2

El archivo A01793101_A4.2.zip cuenta con el presente documento, así como el código fuente y los resultados de la ejecución para los ejercicios de programación de la actividad 4.2.

Los archivos de código fuente se encuentran en el folder de inicio y los resultados se encuentran distribuidos en folders de acuerdo con el número de ejercicio (P1, P2, P3).



A continuación, se adjuntan las evidencias de ejecución de pylint en primera instancia, con los errores que fueron arrojados al ejecutar por primera vez el análisis del código. Así mismo se adjunta la ejecución del análisis ya realizadas las correcciones adecuadas.

PROBLEMA 1: Compute statistics

Nota: Para poder cumplir con el estándar de PEP-8 se ha cambiado el nombre del archivo para cumplir con el camel case a compute_statistics.py

Código Fuente

compute_statistics.py

A01793101_PruebasDeSoftware > A01793101_A4.2 > compute_statistics.py > ...

You, hace 23 horas | 1 author (You)

```python
"""       You, hace 23 horas • First working changes
    Program to calculate all descriptive statistics
    The descriptive statistics are mean, median,
    mode, standard deviation, and variance.
"""
import sys
import os
import re
import time

def median(numbers_list):
    """Function to calculate the median of a list of numbers."""
    # Calcular el índice central.
    mid = len(numbers_list) // 2

    # Si el número de elementos es par, tomar el promedio
    # de los dos valores centrales.
    if len(numbers_list) % 2 == 0:
        return (sorted(numbers_list)[mid] + sorted(numbers_list)[~mid]) / 2
    # Si es impar, el valor central es la mediana.
    return numbers_list[mid]

def calculate_mode(numbers_list):
    """Function to calculate the mode of a list of numbers."""
    numbers_map = {}
    mode = numbers_list[0]
    for element in numbers_list:
        if element in numbers_map:
            numbers_map[element] += 1
            if numbers_map[element] > numbers_map[mode]:
                mode = element
        else:
            numbers_map[element] = 1
    return mode

def variance(numbers_sum, numbers_list):
    """Function to calculate the variance of a list of numbers."""
    mean = numbers_sum / len(numbers_list)
    # Calcular la suma de los cuadrados de las diferencias entre cada elemento y la media.
    dif_sum_square = sum((list_element - mean) ** 2 for list_element in numbers_list)
    # Regresar la Varianza.
    return dif_sum_square / len(numbers_list)
```

compute_statistics.py

A01793101_PruebasDeSoftware > A01793101_A4.2 > compute_statistics.py > ...

```python
def calculate_sd(numbers_sum, numbers_list):
    """Function to calculate the standar deviation of a list of numbers."""
    var = variance(numbers_sum, numbers_list)
    return var ** 0.5

def main():
    """Main program function definition."""
    start_time = time.time()
    with open(sys.argv[1], 'r', encoding="UTF-8") as test_file:
        contents = test_file.readlines()

    list_of_numbers = []
    sum_of_numbers = 0
    i = 0
    test_case_key = f"{re.split(r'(TC[0-9])',sys.argv[1])[1]}"
    for line in contents:
        try:
            list_of_numbers.append(float(line))
            sum_of_numbers += list_of_numbers[i]
            i += 1
        except ValueError:
            print(f"{line} is not numeric type")

    mediana = median(list_of_numbers)
    moda = calculate_mode(list_of_numbers)
    desv_standar = calculate_sd(sum_of_numbers, list_of_numbers)
    varianza = variance(sum_of_numbers, list_of_numbers)

    r_file_path = f"{re.split(r'(TC[0-9])',sys.argv[1])[0]}\\StatisticsResults_{test_case_key}.txt"
    if os.path.isfile(r_file_path): # Clean all file content
        with open(r_file_path, 'w', encoding="UTF-8") as result_file:
            result_file.close()

    with open(r_file_path, 'a', encoding="UTF-8") as result_file:
        result_file.write(f"{test_case_key}\n")
        result_file.write(f"COUNT: {len(list_of_numbers)}\n")
        result_file.write(f"MEDIA: {sum_of_numbers / len(list_of_numbers):.5f}\n")
        result_file.write(f"MEDIANA: {mediana}\n")
        result_file.write(f"MODA: {moda}\n")
        result_file.write(f"SD: {desv_standar:.5f}\n")
        result_file.write(f"VARIANZA: {varianza:.5f}\n\n")

    print(f"COUNT: {len(list_of_numbers)}")
    print(f"MEDIA: {sum_of_numbers / len(list_of_numbers):.5f}")
    print(f"MEDIANA: {mediana}")
    print(f"MODA: {moda}")
    print(f"SD: {desv_standar:.5f}")
    print(f"VARIANZA: {varianza:.5f}")
```

Primer intento de ejecución en pylint:

```
(base) C:\Users\gmuni\OneDrive\Documentos\TEC_Maestria\Pruebas de Software\A01793102_A4.2>pylint computeStatistics.py
************* Module computeStatistics
computeStatistics.py:14:0: C0303: Trailing whitespace (trailing-whitespace)
computeStatistics.py:37:0: C0303: Trailing whitespace (trailing-whitespace)
computeStatistics.py:38:0: C0303: Trailing whitespace (trailing-whitespace)
computeStatistics.py:52:0: C0303: Trailing whitespace (trailing-whitespace)
computeStatistics.py:1:0: C0114: Missing module docstring (missing-module-docstring)
computeStatistics.py:1:0: C0103: Module name "computeStatistics" doesn't conform to snake_case naming style (invalid-nam
e)
computeStatistics.py:3:0: C0116: Missing function or method docstring (missing-function-docstring)
computeStatistics.py:9:4: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (
no-else-return)
computeStatistics.py:15:0: C0116: Missing function or method docstring (missing-function-docstring)
computeStatistics.py:17:4: W0621: Redefining name 'mode' from outer scope (line 15) (redefined-outer-name)
computeStatistics.py:27:0: C0116: Missing function or method docstring (missing-function-docstring)
computeStatistics.py:34:0: C0116: Missing function or method docstring (missing-function-docstring)
computeStatistics.py:34:0: C0103: Function name "sd" doesn't conform to snake_case naming style (invalid-name)
computeStatistics.py:39:5: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
computeStatistics.py:43:0: C0103: Constant name "sum_of_numbers" doesn't conform to UPPER_CASE naming style (invalid-nam
e)

-----------------------------------
Your code has been rated at 6.34/10


(base) C:\Users\gmuni\OneDrive\Documentos\TEC_Maestria\Pruebas de Software\A01793102_A4.2>
```

Ejecución de Pylint con calificación 10/10:

```
(base) C:\Users\gmuni\OneDrive\Documentos\TEC_Maestria\Pruebas de Software\A01793102_A4.2>pylint compute_Statistics.py

--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.76/10, +0.24)


(base) C:\Users\gmuni\OneDrive\Documentos\TEC_Maestria\Pruebas de Software\A01793102_A4.2>
```

Problema 2: Convert numbers

Nota: Para poder cumplir con el estándar de PEP-8 se ha cambiado el nombre del archivo para cumplir con el camel case a convert_numbers.py

Código Fuente

convert_numbers.py

A01793101_PruebasDeSoftware > A01793101_A4.2 > convert_numbers.py > ...

You, hace 23 horas | 1 author (You)

```python
"""       You, hace 23 horas • First working changes
    Program to convert numbers to binary and hexadecimal base.
    The results shall be print on a screen and on a file named
    ConvertionResults.txt.
"""
import sys
import os
import re
import time
import copy

def decimal_a_binario(numero_decimal, bit_width=8):
    """Function para convertir decimales a binarios."""
    if numero_decimal < 0:
        numero_decimal = 2**bit_width + numero_decimal

    num_binario = ""

    while numero_decimal > 0:
        resto = numero_decimal % 2
        num_binario = str(resto) + num_binario
        numero_decimal = numero_decimal // 2

    # Rellenar con ceros a la izquierda para alcanzar el ancho de bits deseado
    while len(num_binario) < bit_width:
        num_binario = "0" + num_binario

    return num_binario

def decimal_a_hexadecimal(numero_decimal, bit_width=8):
    """Function para convertir decimales a hexadecimales."""
    my_decimal = copy.copy(numero_decimal)
    if numero_decimal < 0:
        numero_decimal = (1 << bit_width) + numero_decimal

    hexadecimal_result = ""

    while numero_decimal > 0:
        restante = numero_decimal % 16
        # Convertir los restos mayores a 7 a letras (A-F)
        if restante < 10:
            hexadecimal_result = str(restante) + hexadecimal_result
        else:
            hexadecimal_result = chr(ord('A') + restante - 10) + hexadecimal_result
        numero_decimal = numero_decimal // 16
```

convert_numbers.py

A01793101_PruebasDeSoftware > A01793101_A4.2 > convert_numbers.py > ...

```python
    # Rellenar con 'F' a la izquierda en numeros negativos
    if my_decimal < 0:
        while len(hexadecimal_result) < bit_width:
            hexadecimal_result = "F" + hexadecimal_result
    elif my_decimal == 0:
        hexadecimal_result = "0"

    return hexadecimal_result.upper()

def main():
    """Main program function definition."""
    start_time = time.time()
    with open(sys.argv[1], 'r', encoding="UTF-8") as test_file:
        contents = test_file.readlines()

    test_case_key = f"{re.split(r'(TC[0-9])',sys.argv[1])[1]}"
    r_file_path = f"{re.split(r'(TC[0-9])',sys.argv[1])[0]}\\ConvertionResults_{test_case_key}.txt"
    # If file exists, delete its content first.
    if os.path.isfile(r_file_path):
        with open(r_file_path, 'w', encoding="UTF-8") as result_file:
            result_file.close()
    with open(r_file_path, 'w', encoding="UTF-8") as result_file:
        # Crear el archivo with line of headers
        result_file.write(f"{test_case_key}\n")
        result_file.write("Numero Original\t\t\tNumero Binario\t\t\tNumero Hexadecimal\n")

    print("Numero Original\t\tNumero Binario\t\tNumero Hexadecimal")
    for line in contents:
        try:
            binario = decimal_a_binario(int(line))
            hexadecimal = decimal_a_hexadecimal(int(line))
            print(f"{int(line)}\t\t\t{binario}\t\t\t{hexadecimal}")
            with open(r_file_path, 'a', encoding="UTF-8") as result_file:
                result_file.write(f"\t{int(line)}\t\t\t{binario}\t\t\t{hexadecimal}\n")
        except ValueError:
            print(f"{line} is not numeric type")

    end_time = time.time()
    print(f"\nTiempo de ejecución total: {end_time - start_time:.5f} segundos")

    # Agregar el tiempo de ejecución al final del archivo de resultados
    with open(r_file_path, 'r', encoding="UTF-8") as result_file:
        contents = result_file.readlines()

    with open(r_file_path, 'w', encoding="UTF-8") as result_file:
        for line in contents[:-1]:  # Copiar todo excepto la última línea
            result_file.write(line)
        result_file.write(f"Tiempo de ultima ejecucion: {end_time - start_time:.5f}\n")
```

Primer intento de ejecución de pylint

```
(base) C:\Users\gmuni\OneDrive\Documentos\TEC_Maestria\Pruebas de Software\A01793102_A4.2>pylint convert_numbers.py
************* Module convert_numbers
convert_numbers.py:3:62: C0303: Trailing whitespace (trailing-whitespace)
convert_numbers.py:48:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
convert_numbers.py:62:4: W0612: Unused variable 'test_case_key' (unused-variable)
convert_numbers.py:7:0: W0611: Unused import os (unused-import)

-----------------------------------
Your code has been rated at 9.20/10
```

Ejecución de pylint con calificación de 10/10

```
(base) C:\Users\gmuni\OneDrive\Documentos\TEC_Maestria\Pruebas de Software\A01793102_A4.2>pylint convert_numbers.py

-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.20/10, +0.80)
```

Problema 3: Word count

Nota: Para poder cumplir con el estándar de PEP-8 se ha cambiado el nombre del archivo para cumplir con el camel case a Word_count.py

Código Fuente

word_count.py

A01793101_PruebasDeSoftware > A01793101_A4.2 > word_count.py > ...

You, hace 23 horas | 1 author (You)

```python
"""       You, hace 23 horas • First working changes
    Program to  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
"""
import sys
import os
import re
import time

def main():
    """Main program function definition."""
    start_time = time.time()
    with open(sys.argv[1], 'r', encoding="UTF-8") as test_file:
        contents = test_file.readlines()

    test_case_key = f"{re.split(r'(TC[0-9])',sys.argv[1])[1]}"
    r_file_path = f"{re.split(r'(TC[0-9])',sys.argv[1])[0]}\\WordCountResults_{test_case_key}.txt"
    if os.path.isfile(r_file_path):
        with open(r_file_path, 'w', encoding="UTF-8") as result_file:
            result_file.close()
    with open(r_file_path, 'w', encoding="UTF-8") as result_file:
        # Crear el archivo with line of headers
        result_file.write(f"{test_case_key}\n")
        result_file.write("Palabra\t\t\tNumero de apariciones\t\t\t\n")

    print("Palabra\t\tNumero de apariciones\t\t\n")
    mapa_de_palabras = {}
    for line in contents:
        try:
            palabra = line.strip()
            if palabra in mapa_de_palabras:
                mapa_de_palabras[palabra] += 1
            else:
                mapa_de_palabras[palabra] = 1
        except ValueError:
            print(f"{line} is not string type")

    mapa_de_palabras = dict(sorted(mapa_de_palabras.items(), key=lambda x:x[1], reverse=True))
    for palabra, num_de_apariciones in mapa_de_palabras.items():
        print(f"{palabra}\t\t{num_de_apariciones}\n")
        with open(r_file_path, 'a', encoding="UTF-8") as result_file:
            result_file.write(f"{palabra}\t\t\t{num_de_apariciones}\n")

    end_time = time.time()
    print(f"\nTiempo de ejecución total: {end_time - start_time:.5f} segundos")
```

Primer intento de ejecución de pylint

```
(base) C:\Users\gmuni\OneDrive\Documentos\TEC_Maestria\Pruebas de Software\A01793102_A4.2>pylint word_count.py
************* Module word_count
word_count.py:2:51: C0303: Trailing whitespace (trailing-whitespace)
word_count.py:4:60: C0303: Trailing whitespace (trailing-whitespace)
word_count.py:20:0: C0301: Line too long (104/100) (line-too-long)
word_count.py:39:0: C0303: Trailing whitespace (trailing-whitespace)
word_count.py:44:0: W0311: Bad indentation. Found 16 spaces, expected 12 (bad-indentation)
word_count.py:22:8: R1732: Consider using 'with' for resource-allocating operations (consider-using-with)
word_count.py:22:8: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
word_count.py:33:31: C0201: Consider iterating the dictionary directly instead of calling .keys() (consider-iterating-dictionary)
word_count.py:11:0: W0611: Unused import copy (unused-import)

-----------------------------------
Your code has been rated at 7.80/10


(base) C:\Users\gmuni\OneDrive\Documentos\TEC_Maestria\Pruebas de Software\A01793102_A4.2>
```

Ejecución de pylint con calificación de 10/10

```
(base) C:\Users\gmuni\OneDrive\Documentos\TEC_Maestria\Pruebas de Software\A01793102_A4.2>pylint word_count.py
************* Module word_count
word_count.py:33:26: C0201: Consider iterating the dictionary directly instead of calling .keys() (consider-iterating-dictionary)

--------------------------------------------------------------------
Your code has been rated at 9.76/10 (previous run: 7.80/10, +1.95)


(base) C:\Users\gmuni\OneDrive\Documentos\TEC_Maestria\Pruebas de Software\A01793102_A4.2>pylint word_count.py

--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.76/10, +0.24)
```