Materia:

Pruebas de Software y Aseguramiento de la Calidad

Actividad:

Ejercicio de programación 1

Alumno:

José Antonio Toledo González - A01796592

Profesor titular:

Dr. Gerardo Padilla Zárate

Profesor tutor:

Gabriela Hernández

Fecha de entrega:

08 de febrero de 2026

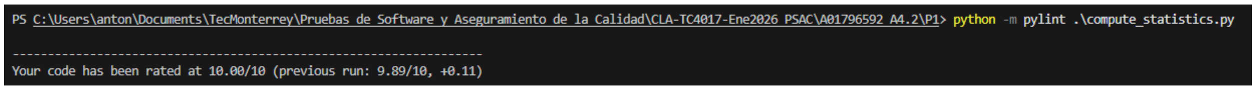
Problemas

1. Compute statistics

Source

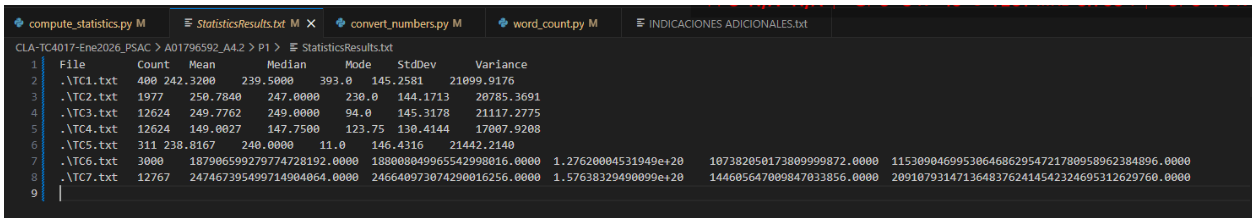
```
""" Compute Statistics

CLA-T-1201 - A01796592 - Assignment 4.2 - Problem 1

This script computes the mean, median, mode, standard deviation , and variance of a list of numbers.

"""


import sys

import time


class Statistics:

    """Class to compute statistical measures of a list of numbers."""

    def __init__(self, arg_numbers):

        self.numbers = arg_numbers


    def mean_calc(self):

        """Calculate the mean of the numbers."""

        mean = 0

        try:

            self.validate_numbers()

            for num in self.numbers:

                mean += num

            mean /= len(self.numbers)

        except ZeroDivisionError as exc:

            raise ValueError('The list of numbers is empty.') from exc

        return mean


    def median(self):

        """Calculate the median of the numbers."""

        median = 0

        sorted_numbers = self.numbers.copy()

        n = len(sorted_numbers)

        try:

            self.validate_numbers()

            for i in range(n):
                for j in range(0, n-i-1):
                    if sorted_numbers[j] > sorted_numbers[j+1]:
                        temp = sorted_numbers[j]
                        sorted_numbers[j] = sorted_numbers[j+1]
                        sorted_numbers[j+1] = temp
            if n % 2 == 0:
                index = int (n/2)
                median = (sorted_numbers[index - 1] + sorted_numbers[index]) / 2
            else:
                median = sorted_numbers[int(n/2)]
        except ZeroDivisionError as exc:
            raise ValueError('The list of numbers is empty.') from exc
        return median

    def mode(self):
        """Calculate the mode of the numbers."""
        frequency = {}
        mode = None
        max_freq = 0
        try:
            self.validate_numbers()
            for num in self.numbers:
                for i, val in enumerate(self.numbers):
                    if num == val:
                        frequency[num] = frequency.get(num, 0) + 1
            for i, num in enumerate(self.numbers):
                if frequency[self.numbers[i]] > max_freq:
                    max_freq = frequency[self.numbers[i]]
                    mode = self.numbers[i]
        except ZeroDivisionError as exc:
            raise ValueError('The list of numbers is empty.') from exc
        return mode

    def variance(self):
        """Calculate the variance of the numbers."""
        mean = self.mean_calc()
        variance = 0
        try:
            self.validate_numbers()
            for num in self.numbers:
                variance += (abs(num - mean) ** 2)
            variance /= len(self.numbers)
        except ZeroDivisionError as exc:
            raise ValueError('The list of numbers is empty.') from exc
        return variance

    def standard_deviation(self):
        """Calculate the standard deviation of the numbers."""
        variance = self.variance()
        std_dev = variance ** 0.5
        return std_dev

    def validate_numbers(self):
        """Validate that all elements in the list are numbers."""
        for num in self.numbers:
            if not isinstance(num, (int, float)):
                raise ValueError("All elements must be numbers.")

if __name__ == "__main__":
    start_time = time.time()
    filenames = sys.argv[1:]
    with open("StatisticsResults.txt", "w", encoding="utf-8") as out:
        out.write(
            "File\t\tCount\tMean\t\tMedian\t\tMode\tStdDev\t\tVariance\n"
        )
        print("File\t\tCount\tMean\t\tMedian\t\tMode\tStdDev\t\tVariance")
        for fname in filenames:
            numbers = []
            with open(fname, "r", encoding="utf-8") as file:
                content = file.read().replace(",", " ").replace("\n", " ").replace(";", " ").split()
                for value in content:
                    try:
                        numbers.append(float(value))
                    except ValueError:
                        print(f"Warning: '{value}' in {fname} ignored")
            stats = Statistics(numbers)
        out.write(
            f"{fname}\t"
            f"{len(numbers)}\t"
            f"{stats.mean_calc():.4f}\t"
            f"{stats.median():.4f}\t"
            f"{stats.mode()}\t"
            f"{stats.standard_deviation():.4f}\t"
            f"{stats.variance():.4f}\n"
        )
        print(
            f"{fname}\t"
            f"{len(numbers)}\t"
            f"{stats.mean_calc():.4f}\t"
            f"{stats.median():.4f}\t"
            f"{stats.mode()}\t"
            f"{stats.standard_deviation():.4f}\t"
            f"{stats.variance():.4f}"
        )
    print("StatisticsResults.txt generated successfully")
    end_time = time.time()
    print(f"Execution time: {end_time - start_time:.6f} seconds")
```

Test



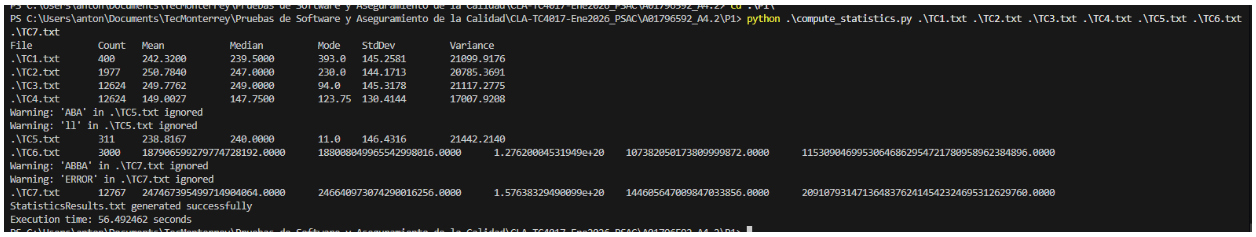
```
PS C:\Users\anton\Documents\TecMonterrey\Pruebas de Software y Aseguramiento de la Calidad\CLA-TC4017-Ene2026 PSAC\A01796592 A4.2\P1> python -m pylint .\compute_statistics.py

------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.89/10, +0.11)
```

Result



compute_statistics.py M
StatisticsResults.txt M
convert_numbers.py M
word_count.py M
INDICACIONES ADICIONALES.txt
CLA-TC4017-Ene2026_PSAC > A01796592_A4.2 > P1 > StatisticsResults.txt

```
File        Count   Mean        Median      Mode    StdDev      Variance
.\TC1.txt   400 242.3200    239.5000    393.0   145.2581    21099.9176
.\TC2.txt   1977    250.7840    247.0000    230.0   144.1713    20785.3691
.\TC3.txt   12624   249.7762    249.0000    94.0    145.3178    21117.2775
.\TC4.txt   12624   149.0027    147.7500    123.75  130.4144    17007.9208
.\TC5.txt   311 238.8167    240.0000    11.0    146.4316    21442.2140
.\TC6.txt   3000    187906599279774728192.0000  188008049965542998016.0000  1.27620004531949e+20    107382050173809999872.0000  11530904699530646862954721780958962384896.0000
.\TC7.txt   12767   247467395499714904064.0000  246640973074290016256.0000  1.57638329490099e+20    144605647009847033856.0000  20910793147136483762414542324695312629760.0000
```



```
PS C:\Users\anton\Documents\TecMonterrey\Pruebas de Software y Aseguramiento de la Calidad\CLA-TC4017-Ene2026_PSAC\A01796592_A4.2> cd .\P1\
PS C:\Users\anton\Documents\TecMonterrey\Pruebas de Software y Aseguramiento de la Calidad\CLA-TC4017-Ene2026_PSAC\A01796592_A4.2\P1> python .\compute_statistics.py .\TC1.txt .\TC2.txt .\TC3.txt .\TC4.txt .\TC5.txt .\TC6.txt
.\TC7.txt
File            Count   Mean            Median          Mode    StdDev          Variance
.\TC1.txt       400     242.3200        239.5000        393.0   145.2581        21099.9176
.\TC2.txt       1977    250.7840        247.0000        230.0   144.1713        20785.3691
.\TC3.txt       12624   249.7762        249.0000        94.0    145.3178        21117.2775
.\TC4.txt       12624   149.0027        147.7500        123.75  130.4144        17007.9208
Warning: 'ABA' in .\TC5.txt ignored
Warning: 'll' in .\TC5.txt ignored
.\TC5.txt       311     238.8167        240.0000        11.0    146.4316        21442.2140
.\TC6.txt       3000    187906599279774728192.0000      188008049965542998016.0000      1.27620004531949e+20    107382050173809999872.0000      11530904699530646862954721780958962384896.0000
Warning: 'ABBA' in .\TC7.txt ignored
Warning: 'ERROR' in .\TC7.txt ignored
.\TC7.txt       12767   247467395499714904064.0000      246640973074290016256.0000      1.57638329490099e+20    144605647009847033856.0000      20910793147136483762414542324695312629760.0000
StatisticsResults.txt generated successfully
Execution time: 56.492462 seconds
PS C:\Users\anton\Documents\TecMonterrey\Pruebas de Software y Aseguramiento de la Calidad\CLA-TC4017-Ene2026_PSAC\A01796592_A4.2\P1>
```

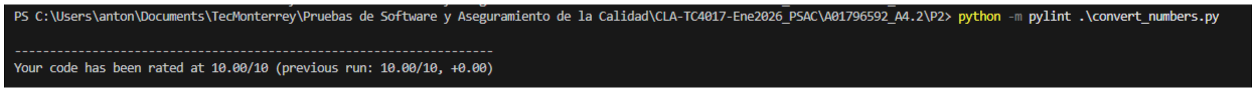
2. Converter

Source

```
""" Convert numbers

CLA-T-1201 - A01796592 - Assignment 4.2 - Problem 2

This script reads a file containing numbers separated by commas, spaces, semicolons, or new lines,

and converts them to binary, and hexadecimal formats, saving the results to separate files.

"""


import sys

import time


class NumberConverter:

    """Class to convert numbers to different formats."""

    def __init__(self, arg_numbers):

        self.numbers = arg_numbers


    def to_binary(self):

        """Convert numbers to binary format."""

        binary_numbers = []

        for num_to_bin in self.numbers:

            try:

                n = int(num_to_bin)

                binary = ""

                if n == 0:

                    binary = "0"

                elif n < 0:

                    n = abs(n)

                    while n > 0:

                        binary = str(n % 2) + binary

                        n //= 2

                    binary = binary.zfill(10)

                    binary = ''.join('1' if b == '0' else '0' for b in binary)

                    binary = format(int(binary, 2) + 1, f'0{10}b')

                else:
```

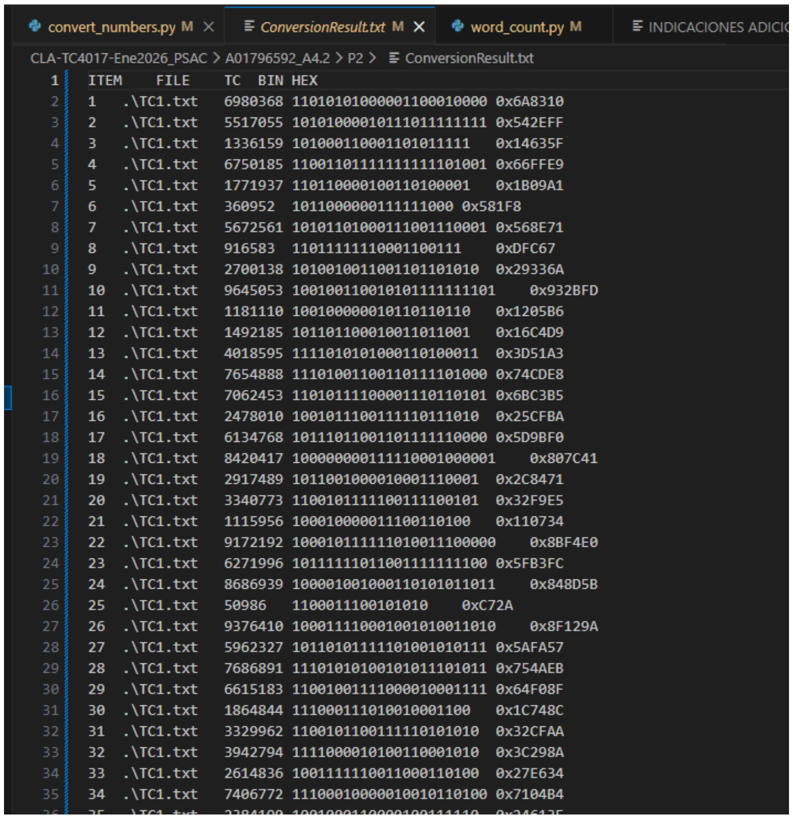
```
                while n > 0:
                    binary = str(n % 2) + binary
                    n //= 2
            binary_numbers.append(binary)
        except ValueError:
            print(f"Warning: '{num_to_bin}' is not a valid integer and will be skipped.")
    return binary_numbers

def to_hexadecimal(self):
    """Convert numbers to hexadecimal format."""
    hex_numbers = []
    for num_to_hex in self.numbers:
        try:
            n = int(num_to_hex)
            hex_digits = "0123456789ABCDEF"
            hexadecimal = ""
            if n == 0:
                hexadecimal = "0"
            elif n < 0:
                n = abs(n)
                while n > 0:
                    hexadecimal = hex_digits[n % 16] + hexadecimal
                    n //= 16
                hexadecimal = hexadecimal.zfill(10)
                hexadecimal = ''.join(hex_digits[15 - hex_digits.index(h)] for h in hexadecimal)
                hexadecimal = format(int(hexadecimal, 16) + 1, f'0{10}X')
                hexadecimal = "0x" + hexadecimal
            else:
                while n > 0:
                    hexadecimal = hex_digits[n % 16] + hexadecimal
                    n //= 16
                hexadecimal = "0x" + hexadecimal
            hex_numbers.append(hexadecimal)
        except ValueError:
            print(f"Warning: '{num_to_hex}' is not a valid integer and will be skipped.")
        return hex_numbers


if __name__ == "__main__":
    start_time = time.time()
    filenames = sys.argv[1:]
    with open("ConversionResult.txt", "w", encoding="utf-8") as out:
        out.write("ITEM\tFILE\tTC\tBIN\tHEX\n")
        print("ITEM\tFILE\tTC\tBIN\tHEX")
        for fname in filenames:
            item = 1
            numbers = []
            with open(fname, 'r', encoding='utf-8') as file:
                tokens = (
                    file.read()
                    .replace(",", " ")
                    .replace("\n", " ")
                    .replace(";", " ")
                    .split()
                )
                for num in tokens:
                    try:
                        numbers.append(int(num))
                    except ValueError:
                        print(f"Warning: '{num}' in {fname} skipped")
            converter = NumberConverter(numbers)
            bin_nums = converter.to_binary()
            hex_nums = converter.to_hexadecimal()
            for i, num in enumerate(numbers):
                out.write(
                    f"{item}\t{fname}\t{num}\t{bin_nums[i]}\t{hex_nums[i]}\n"
                )
                print(
                    f"{item}\t{fname}\t{num}\t{bin_nums[i]}\t{hex_nums[i]}"
                )
                item += 1
            out.write("\n")

            print("\n")

    end_time = time.time()

    print(f"\nExecution time: {end_time - start_time:.6f} seconds")
```

Test

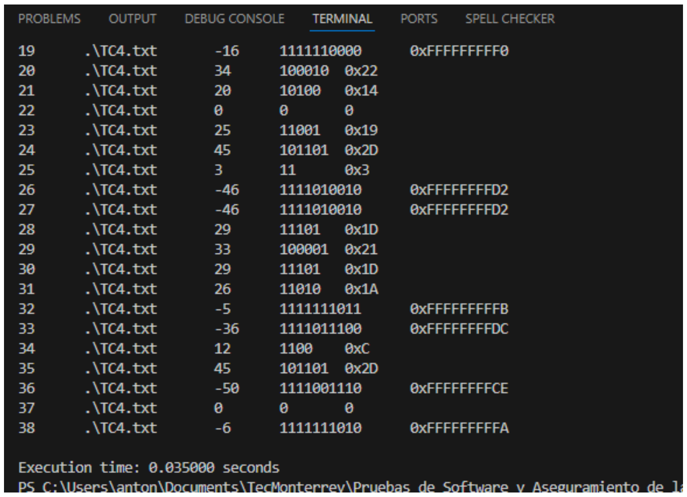


```
PS C:\Users\anton\Documents\TecMonterrey\Pruebas de Software y Aseguramiento de la Calidad\CLA-TC4017-Ene2026_PSAC\A01796592_A4.2\P2> python -m pylint .\convert_numbers.py

------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
```

Result



```
ITEM    FILE      TC   BIN HEX
1   .\TC1.txt   6980368 11010101000001100010000 0x6A8310
2   .\TC1.txt   5517055 10101000010111011111111 0x542EFF
3   .\TC1.txt   1336159 101000110001101011111   0x14635F
4   .\TC1.txt   6750185 11001101111111111101001 0x66FFE9
5   .\TC1.txt   1771937 110110000100110100001   0x1B09A1
6   .\TC1.txt   360952  1011000000111111000 0x581F8
7   .\TC1.txt   5672561 10101101000111001110001 0x568E71
8   .\TC1.txt   916583  11011111110001100111    0xDFC67
9   .\TC1.txt   2700138 1010010011001101101010  0x29336A
10  .\TC1.txt   9645053 100100110010101111111101    0x932BFD
11  .\TC1.txt   1181110 100100000010110110110   0x1205B6
12  .\TC1.txt   1492185 101101100010011011001   0x16C4D9
13  .\TC1.txt   4018595 1111010101000110100011  0x3D51A3
14  .\TC1.txt   7654888 11101001100110111101000 0x74CDE8
15  .\TC1.txt   7062453 11010111100001110110101 0x6BC3B5
16  .\TC1.txt   2478010 1001011100111110111010  0x25CFBA
17  .\TC1.txt   6134768 10111011001101111110000 0x5D9BF0
18  .\TC1.txt   8420417 100000000111110001000001    0x807C41
19  .\TC1.txt   2917489 1011001000010001110001  0x2C8471
20  .\TC1.txt   3340773 1100101111100111100101  0x32F9E5
21  .\TC1.txt   1115956 100010000011100110100   0x110734
22  .\TC1.txt   9172192 100010111111010011100000    0x8BF4E0
23  .\TC1.txt   6271996 10111111011001111111100 0x5FB3FC
24  .\TC1.txt   8686939 100001001000110101011011    0x848D5B
25  .\TC1.txt   50986   1100011100101010    0xC72A
26  .\TC1.txt   9376410 100011110001001010011010    0x8F129A
27  .\TC1.txt   5962327 10110101111101001010111 0x5AFA57
28  .\TC1.txt   7686891 11101010100101011101011 0x754AEB
29  .\TC1.txt   6615183 11001001111000010001111 0x64F08F
30  .\TC1.txt   1864844 111000111010010001100   0x1C748C
31  .\TC1.txt   3329962 1100101100111110101010  0x32CFAA
32  .\TC1.txt   3942794 1111000010100110001010  0x3C298A
33  .\TC1.txt   2614836 1001111110011000110100  0x27E634
34  .\TC1.txt   7406772 11100010000010010110100 0x7104B4
```



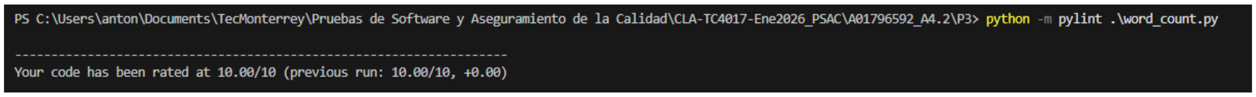
```
19      .\TC4.txt       -16     1111110000      0xFFFFFFFFF0
20      .\TC4.txt       34      100010  0x22
21      .\TC4.txt       20      10100   0x14
22      .\TC4.txt       0       0       0
23      .\TC4.txt       25      11001   0x19
24      .\TC4.txt       45      101101  0x2D
25      .\TC4.txt       3       11      0x3
26      .\TC4.txt       -46     1111010010      0xFFFFFFFFD2
27      .\TC4.txt       -46     1111010010      0xFFFFFFFFD2
28      .\TC4.txt       29      11101   0x1D
29      .\TC4.txt       33      100001  0x21
30      .\TC4.txt       29      11101   0x1D
31      .\TC4.txt       26      11010   0x1A
32      .\TC4.txt       -5      1111111011      0xFFFFFFFFFB
33      .\TC4.txt       -36     1111011100      0xFFFFFFFFDC
34      .\TC4.txt       12      1100    0xC
35      .\TC4.txt       45      101101  0x2D
36      .\TC4.txt       -50     1111001110      0xFFFFFFFFCE
37      .\TC4.txt       0       0       0
38      .\TC4.txt       -6      1111111010      0xFFFFFFFFFA

Execution time: 0.035000 seconds
PS C:\Users\anton\Documents\TecMonterrey\Pruebas de Software y Aseguramiento de la
```

3. Count Words

Source

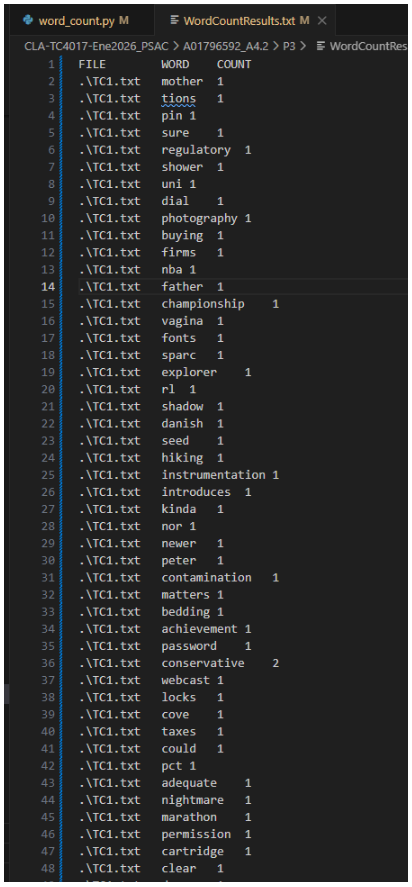
```
""" Word Count

CLA-T-1201 - A01796592 - Assignment 4.2 - Problem 2

This script identify all distintic words and the frequency of them

"""


import sys

import time


class WordCount:

    """ Word Count class

    This class has the method to count the frequency of words in a text file

    """

    def __init__(self, arg_word_count):

        """ Constructor of the class

        It initializes the array of words and an empty dictionary to store the word count

        """

        self.array_words = arg_word_count

        self.word_count = {}

        self.total_words = 0


    def count_words(self):

        """ Count the frequency of words in the array of words

        It iterates through the array of words and counts the frequency of each word

        """

        for w in self.array_words:

            word_lower = w.lower()

            if word_lower in self.word_count:

                self.word_count[word_lower] += 1

                self.total_words += 1

            else:

                self.word_count[word_lower] = 1

                self.total_words += 1
    def save_to_file(self, output_filename):
        """ Save the word count to a file
        It writes the word and its frequency to the specified output file
        """
        with open(output_filename, 'w', encoding='utf-8') as f:
            for word_write, count in self.word_count.items():
                f.write(f"{word_write}: \t{count}\n")
                print(f"{word_write}: \t{count}")
            f.write(f"Total words: {self.total_words}\n")
            print(f"Total words: {self.total_words}\n")


if __name__ == "__main__":
    start_time = time.time()
    filenames = sys.argv[1:]
    with open("WordCountResults.txt", 'w', encoding='utf-8') as output_file:
        output_file.write("FILE\t\tWORD\tCOUNT\n")
        print("FILE\t\tWORD\tCOUNT")
        for fname in filenames:
            words = []
            with open(fname, 'r', encoding='utf-8') as file:
                tokens = (
                file.read()
                .replace(",", " ")
                .replace("\n", " ")
                .replace(";", " ")
                .split()
            )
            for word in tokens:
                if word.isalpha():
                    words.append(word.lower())
                else:
                    print(f"Warning: '{word}' in {fname} skipped")
            counter = WordCount(words)
            counter.count_words()
            for i, word in enumerate(words):
                output_file.write(f"{fname}\t{word}\t{counter.word_count[word]}\n")
                print(f"{fname}\t{word}\t{counter.word_count[word]}")
            output_file.write(f"{fname}\tTotal words\t{counter.total_words}\n")
            print(f"{fname}\tTotal words\t{counter.total_words}\n")
            output_file.write("\n")
    end_time = time.time()
    print(f"Execution time: {end_time - start_time:.6f} seconds")
```

Test



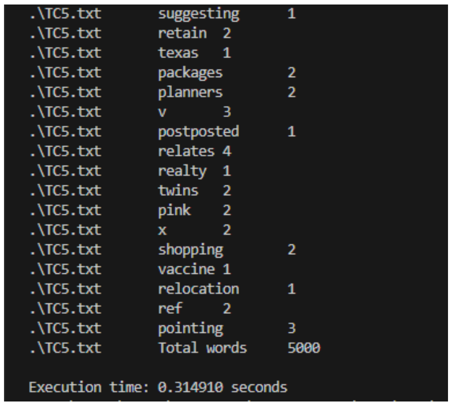
```
PS C:\Users\anton\Documents\TecMonterrey\Pruebas de Software y Aseguramiento de la Calidad\CLA-TC4017-Ene2026_PSAC\A01796592_A4.2\P3> python -m pylint .\word_count.py

------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
```

Result



word_count.py M
WordCountResults.txt M
CLA-TC4017-Ene2026_PSAC > A01796592_A4.2 > P3 > WordCountRes

```
FILE        WORD    COUNT
.\TC1.txt   mother  1
.\TC1.txt   tions   1
.\TC1.txt   pin 1
.\TC1.txt   sure    1
.\TC1.txt   regulatory  1
.\TC1.txt   shower  1
.\TC1.txt   uni 1
.\TC1.txt   dial    1
.\TC1.txt   photography 1
.\TC1.txt   buying  1
.\TC1.txt   firms   1
.\TC1.txt   nba 1
.\TC1.txt   father  1
.\TC1.txt   championship    1
.\TC1.txt   vagina  1
.\TC1.txt   fonts   1
.\TC1.txt   sparc   1
.\TC1.txt   explorer    1
.\TC1.txt   rl  1
.\TC1.txt   shadow  1
.\TC1.txt   danish  1
.\TC1.txt   seed    1
.\TC1.txt   hiking  1
.\TC1.txt   instrumentation 1
.\TC1.txt   introduces  1
.\TC1.txt   kinda   1
.\TC1.txt   nor 1
.\TC1.txt   newer   1
.\TC1.txt   peter   1
.\TC1.txt   contamination   1
.\TC1.txt   matters 1
.\TC1.txt   bedding 1
.\TC1.txt   achievement 1
.\TC1.txt   password    1
.\TC1.txt   conservative    2
.\TC1.txt   webcast 1
.\TC1.txt   locks   1
.\TC1.txt   cove    1
.\TC1.txt   taxes   1
.\TC1.txt   could   1
.\TC1.txt   pct 1
.\TC1.txt   adequate    1
.\TC1.txt   nightmare   1
.\TC1.txt   marathon    1
.\TC1.txt   permission  1
.\TC1.txt   cartridge   1
.\TC1.txt   clear   1
```



```
.\TC5.txt       suggesting      1
.\TC5.txt       retain  2
.\TC5.txt       texas   1
.\TC5.txt       packages        2
.\TC5.txt       planners        2
.\TC5.txt       v       3
.\TC5.txt       postposted      1
.\TC5.txt       relates 4
.\TC5.txt       realty  1
.\TC5.txt       twins   2
.\TC5.txt       pink    2
.\TC5.txt       x       2
.\TC5.txt       shopping        2
.\TC5.txt       vaccine 1
.\TC5.txt       relocation      1
.\TC5.txt       ref     2
.\TC5.txt       pointing        3
.\TC5.txt       Total words     5000

Execution time: 0.314910 seconds
```