Tecnológico de Monterrey

Maestría en Inteligencia Artificial Aplicada

Pruebas de software y aseguramiento de la calidad (Gpo 10)

4.2 Ejercicio de programación 1

Dr. Gerardo Padilla Zárate

Ramon Ariel Ivan Muñoz Corona A01330566

Fecha: 04/02/2024

4.2 Ejercicios de Programación 1

Liga de Repositorio: https://github.com/rmunoz78/A01330566_A4.2

Ejercicio 1 - Compute Statistics

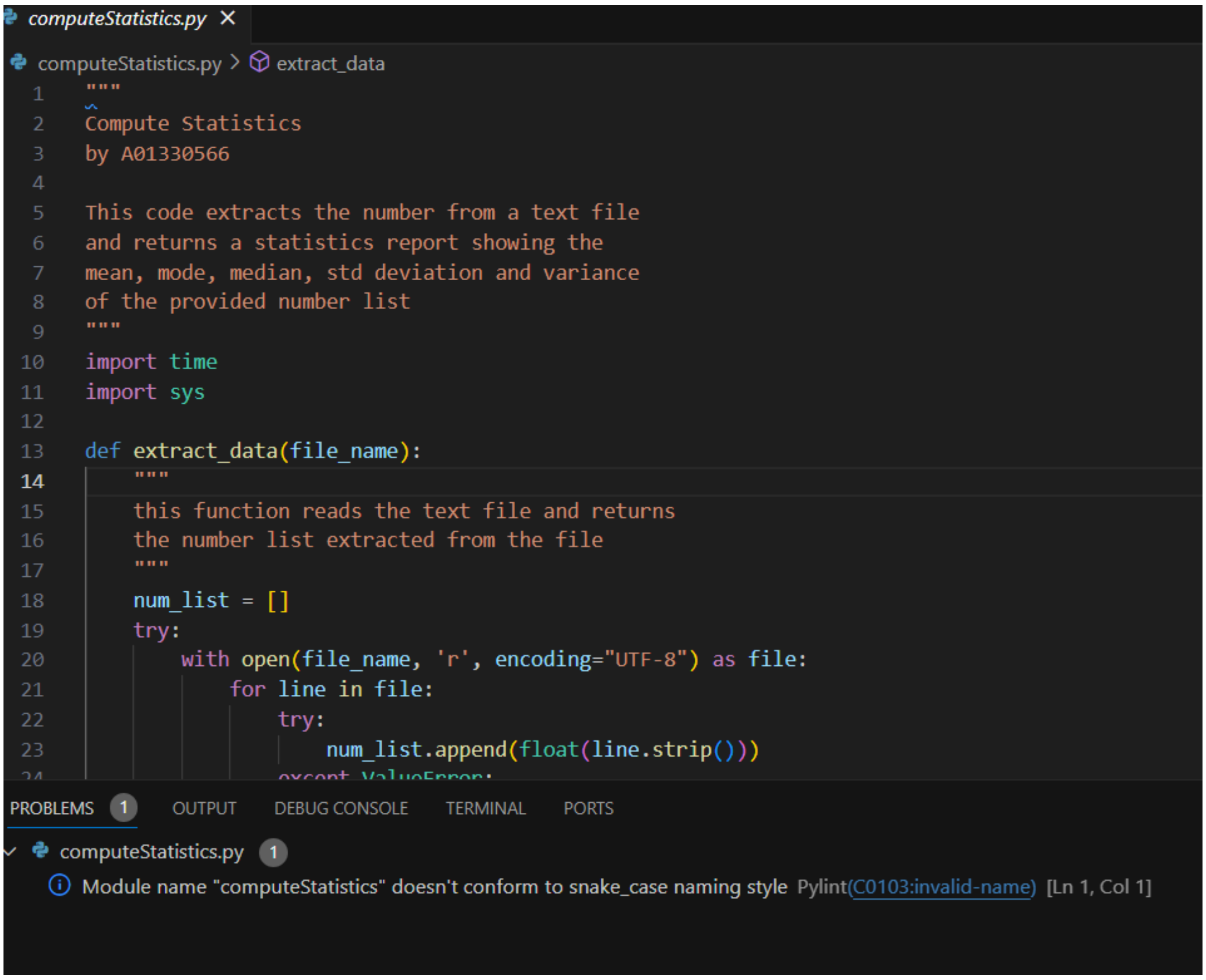
- Req1. The program shall be invoked from a command line. The program shall receive a file as parameter. The file will contain a list of items (presumable numbers).
- Req 2. The program shall compute all descriptive statistics from a file containing numbers. The results shall be print on a screen and 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.
- 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.
- Req 4. The name of the program shall be computeStatistics.py
- Req 5. The minimum format to invoke the program shall be as follows:
  - python computeStatistics.py fileWithData.txt
- Req 6. The program shall manage files having from hundreds of items to thousands of items.
- 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.
- Req 8. Be compliant with PEP8.

Código Fuente

```
"""
Convert Numbers
by A01330566

This code extracts the number from a text file
and returns them converted to binary and hexadecimal
base
"""
import sys
import time

def extract_data(file_name):
    """
    this function reads the text file and returns
    the number list extracted from the file
    """
    num_list = []
    try:
        with open(file_name, 'r', encoding="UTF-8") as file:
            for line in file:
                try:
                    num_list.append(float(line.strip()))
                except ValueError:
                    print("Invalid data found in the file:",
line.strip())
    except FileNotFoundError:
        print("File not found:", file_name)
        sys.exit(1)
    except UnicodeDecodeError:
        print("Error decoding file. Please ensure the file is UTF-8
encoded.")
        sys.exit(1)

    return num_list

def convert_num(num, base):
    """
    this function converts fractional or full numbers to base 16 or 2
    """
    if base == 2:
        digits = "01"
    elif base == 16:
        digits = "0123456789ABCDEF"
    else:
        print("Invalid base. Please verify")

    sign = ""
    if num < 0:
        sign = "-"
        num = num * -1
    int_part = int(num)
    flt_part = num-int_part
    result = ""

    while int_part > 0:
        result = digits[int_part % base] + result
        int_part //= base

    if flt_part > 0:
        flt_conv = ""
        for _ in range(8):
            flt_part *= base
            flt_conv += digits[int(flt_part)]
            flt_part -= int(flt_part)

        result = result + "." + flt_conv

    return sign + result

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python compute_statistics.py InputFile.txt")
        sys.exit(1)

    start_time = time.time()
    filename = sys.argv[1]
    file_data = extract_data(filename)
    new_file_data = []
    for item in file_data:
        new_file_data.append(f"DEC: {item}")
        new_file_data.append(f"HEX: {convert_num(item, 16)}")
        new_file_data.append(f"BIN: {convert_num(item, 2)}")
        new_file_data.append("="*5)
    elapsed_time = time.time() - start_time
    new_file_data.append(f"Time elapsed:{elapsed_time} seconds")
    with open("ConvertionResults.txt", 'w', encoding="UTF-8") as
results_file:
        for new_line in new_file_data:
            print(new_line)
            results_file.write(new_line +"\n")
```

Resultados PyLint



computeStatistics.py > extract_data

```python
"""
Compute Statistics
by A01330566

This code extracts the number from a text file
and returns a statistics report showing the
mean, mode, median, std deviation and variance
of the provided number list
"""
import time
import sys

def extract_data(file_name):
    """
    this function reads the text file and returns
    the number list extracted from the file
    """
    num_list = []
    try:
        with open(file_name, 'r', encoding="UTF-8") as file:
            for line in file:
                try:
                    num_list.append(float(line.strip()))
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
computeStatistics.py 1
  Module name "computeStatistics" doesn't conform to snake_case naming style Pylint(C0103:invalid-name) [Ln 1, Col 1]
```

Ejemplo de Archivo Input

```
-508
851
-773
581
-500
954
-340
-343
-710
751
-32
-856
-135
550
680
-821
-60
-485
```

-961
-984
87
537
976
-612
773
92
981
-376
-98
350
836
411
-218
-20
-864
497
444
50
211
703
-23
-230
-302
-613
-542
-309
-107
214
-426
-636
784
94
97
-186
-945
-373
-181
611
-866
-224
-883
-338
229
-902
987
-735

```
-669
-111
-687
-935
922
882
822
808
382
-391
-763
840
-877
-721
-274
-977
28
-521
762
8
-669
-713
240
212
-5
-930
-254
39
153
72
-480
620
-504
-540
AS
asd
asasfd
1231243
asda
```

Resultado de Ejemplo

```
PS C:\Users\rmuno\OneDrive\Documents\GitHub\A01330566_A4.2> python
computeStatistics.py Numbers.txt
Invalid data found in the file: AS
Invalid data found in the file: asd
Invalid data found in the file: asasfd
Invalid data found in the file: asda
Mean: 12107.18811881188
Median: -111.0
Mode: [-669.0]
Standard Deviation: 121914.98359587483
Variance: 14863263225.18243
Time elapsed: 0.0009999275207519531 seconds
```

Ejercicio 2 - Converter

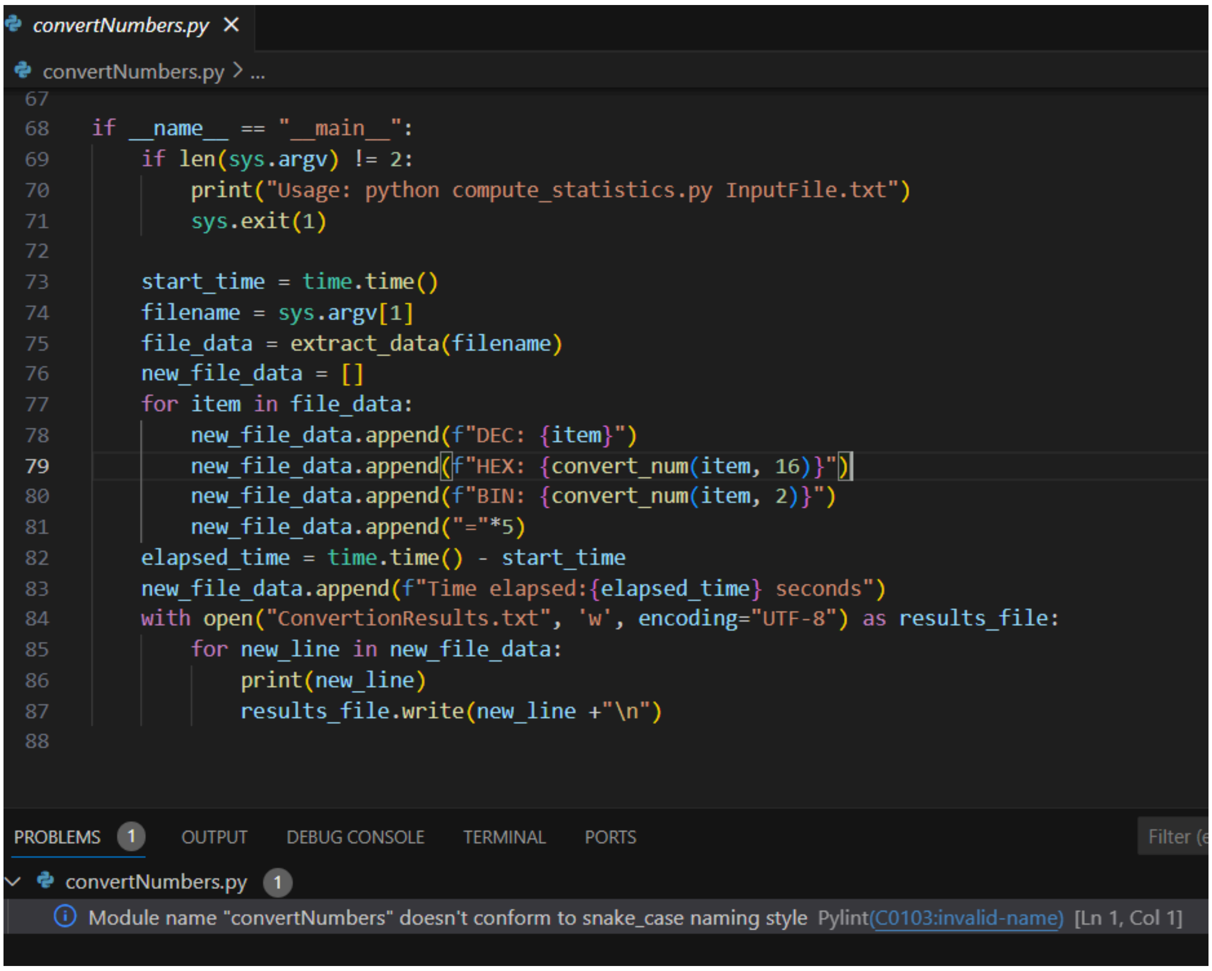
- Req1. The program shall be invoked from a command line. The program shall receive a file as parameter. The file will contain a list of items (presumable numbers).
- Req 2. The program shall convert the numbers to binary and hexadecimal base. The results shall be print on a screen and on a file named ConvertionResults.txt.
  - All computation MUST be calculated using the basic algorithms, not functions or libraries.
- 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.
- Req 4. The name of the program shall be
  - convertNumbers.py
- Req 5. The minimum format to invoke the program shall be as follows:
  - python convertNumbers.py fileWithData.txt
- Req 6. The program shall manage files having from hundreds of items to thousands of items.
- 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.
- Req 8. Be compliant with PEP8.

Código Fuente

```
"""
Convert Numbers
by A01330566

This code extracts the number from a text file
and returns them converted to binary and hexadecimal
base
```

```
"""
import sys
import time

def extract_data(file_name):
    """
    this function reads the text file and returns
    the number list extracted from the file
    """
    num_list = []
    try:
        with open(file_name, 'r', encoding="UTF-8") as file:
            for line in file:
                try:
                    num_list.append(float(line.strip()))
                except ValueError:
                    print("Invalid data found in the file:",
line.strip())
    except FileNotFoundError:
        print("File not found:", file_name)
        sys.exit(1)
    except UnicodeDecodeError:
        print("Error decoding file. Please ensure the file is UTF-8
encoded.")
        sys.exit(1)

    return num_list

def convert_num(num, base):
    """
    this function converts fractional or full numbers to base 16 or 2
    """
    if base == 2:
        digits = "01"
    elif base == 16:
        digits = "0123456789ABCDEF"
    else:
        print("Invalid base. Please verify")

    sign = ""
    if num < 0:
        sign = "-"
        num = num * -1
    int_part = int(num)
    flt_part = num-int_part
    result = ""

    while int_part > 0:
        result = digits[int_part % base] + result
        int_part //= base

    if flt_part > 0:
        flt_conv = ""
        for _ in range(8):
            flt_part *= base
            flt_conv += digits[int(flt_part)]
            flt_part -= int(flt_part)

        result = result + "." + flt_conv

    return sign + result

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python compute_statistics.py InputFile.txt")
        sys.exit(1)

    start_time = time.time()
    filename = sys.argv[1]
    file_data = extract_data(filename)
    new_file_data = []
    for item in file_data:
        new_file_data.append(f"DEC: {item}")
        new_file_data.append(f"HEX: {convert_num(item, 16)}")
        new_file_data.append(f"BIN: {convert_num(item, 2)}")
        new_file_data.append("="*5)
    elapsed_time = time.time() - start_time
    new_file_data.append(f"Time elapsed:{elapsed_time} seconds")
    with open("ConvertionResults.txt", 'w', encoding="UTF-8") as
results_file:
        for new_line in new_file_data:
            print(new_line)
            results_file.write(new_line +"\n")
```

Resultados PyLint



```python
if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python compute_statistics.py InputFile.txt")
        sys.exit(1)

    start_time = time.time()
    filename = sys.argv[1]
    file_data = extract_data(filename)
    new_file_data = []
    for item in file_data:
        new_file_data.append(f"DEC: {item}")
        new_file_data.append(f"HEX: {convert_num(item, 16)}")
        new_file_data.append(f"BIN: {convert_num(item, 2)}")
        new_file_data.append("="*5)
    elapsed_time = time.time() - start_time
    new_file_data.append(f"Time elapsed:{elapsed_time} seconds")
    with open("ConvertionResults.txt", 'w', encoding="UTF-8") as results_file:
        for new_line in new_file_data:
            print(new_line)
            results_file.write(new_line +"\n")
```

```
Module name "convertNumbers" doesn't conform to snake_case naming style Pylint(C0103:invalid-name) [Ln 1, Col 1]
```

Ejemplo de Archivo Input

-508
851
-773
581
-500
954
-340
-343
-710
751
-32
-856
-135
550
680
-821
-60
-485

-961
-984
87
537
976
-612
773
92
981
-376
-98
350
836
411
-218
-20
-864
497
444
50
211
703
-23
-230
-302
-613
-542
-309
-107
214
-426
-636
784
94
97
-186
-945
-373
-181
611
-866
-224
-883
-338
229
-902
987
-735

```
-669
-111
-687
-935
922
882
822
808
382
-391
-763
840
-877
-721
-274
-977
28
-521
762
8
-669
-713
240
212
-5
-930
-254
39
153
72
-480
620
-504
-540
AS
asd
asasfd
1231243
asda
```

Resultado de Ejemplo

```
PS C:\Users\rmuno\OneDrive\Documents\GitHub\A01330566_A4.2> python
convertNumbers.py Numbers.txt

Invalid data found in the file: AS
Invalid data found in the file: asd
Invalid data found in the file: asasfd
Invalid data found in the file: asda
DEC: -508.0
HEX: -1FC
BIN: -111111100
=====
DEC: 851.0
HEX: 353
BIN: 1101010011
=====
DEC: -773.0
HEX: -305
BIN: -1100000101
=====
DEC: 581.0
HEX: 245
BIN: 1001000101
=====
DEC: -500.0
HEX: -1F4
BIN: -111110100
=====
DEC: 954.0
HEX: 3BA
BIN: 1110111010
=====
DEC: -340.0
HEX: -154
BIN: -101010100
=====
DEC: -343.0
HEX: -157
BIN: -101010111
=====
DEC: -710.0
HEX: -2C6
BIN: -1011000110
=====
DEC: 751.0
HEX: 2EF
BIN: 1011101111
=====
DEC: -32.0
HEX: -20
BIN: -100000
=====
DEC: -856.0
HEX: -358
```

```
BIN: -1101011000
=====
DEC: -135.0
HEX: -87
BIN: -10000111
=====
DEC: 550.0
HEX: 226
BIN: 1000100110
=====
DEC: 680.0
HEX: 2A8
BIN: 1010101000
=====
DEC: -821.0
HEX: -335
BIN: -1100110101
=====
DEC: -60.0
HEX: -3C
BIN: -111100
=====
DEC: -485.0
HEX: -1E5
BIN: -111100101
=====
DEC: -961.0
HEX: -3C1
BIN: -1111000001
=====
DEC: -984.0
HEX: -3D8
BIN: -1111011000
=====
DEC: 87.0
HEX: 57
BIN: 1010111
=====
DEC: 537.0
HEX: 219
BIN: 1000011001
=====
DEC: 976.0
HEX: 3D0
BIN: 1111010000
=====
DEC: -612.0
HEX: -264
```

```
BIN: -1001100100
=====
DEC: 773.0
HEX: 305
BIN: 1100000101
=====
DEC: 92.0
HEX: 5C
BIN: 1011100
=====
DEC: 981.0
HEX: 3D5
BIN: 1111010101
=====
DEC: -376.0
HEX: -178
BIN: -101111000
=====
DEC: -98.0
HEX: -62
BIN: -1100010
=====
DEC: 350.0
HEX: 15E
BIN: 101011110
=====
DEC: 836.0
HEX: 344
BIN: 1101000100
=====
DEC: 411.0
HEX: 19B
BIN: 110011011
=====
DEC: -218.0
HEX: -DA
BIN: -11011010
=====
DEC: -20.0
HEX: -14
BIN: -10100
=====
DEC: -864.0
HEX: -360
BIN: -1101100000
=====
DEC: 497.0
HEX: 1F1
```

```
BIN: 111110001
=====
DEC: 444.0
HEX: 1BC
BIN: 110111100
=====
DEC: 50.0
HEX: 32
BIN: 110010
=====
DEC: 211.0
HEX: D3
BIN: 11010011
=====
DEC: 703.0
HEX: 2BF
BIN: 1010111111
=====
DEC: -23.0
HEX: -17
BIN: -10111
=====
DEC: -230.0
HEX: -E6
BIN: -11100110
=====
DEC: -302.0
HEX: -12E
BIN: -100101110
=====
DEC: -613.0
HEX: -265
BIN: -1001100101
=====
DEC: -542.0
HEX: -21E
BIN: -1000011110
=====
DEC: -309.0
HEX: -135
BIN: -100110101
=====
DEC: -107.0
HEX: -6B
BIN: -1101011
=====
DEC: 214.0
HEX: D6
```

```
BIN: 11010110
=====
DEC: -426.0
HEX: -1AA
BIN: -110101010
=====
DEC: -636.0
HEX: -27C
BIN: -1001111100
=====
DEC: 784.0
HEX: 310
BIN: 1100010000
=====
DEC: 94.0
HEX: 5E
BIN: 1011110
=====
DEC: 97.0
HEX: 61
BIN: 1100001
=====
DEC: -186.0
HEX: -BA
BIN: -10111010
=====
DEC: -945.0
HEX: -3B1
BIN: -1110110001
=====
DEC: -373.0
HEX: -175
BIN: -101110101
=====
DEC: -181.0
HEX: -B5
BIN: -10110101
=====
DEC: 611.0
HEX: 263
BIN: 1001100011
=====
DEC: -866.0
HEX: -362
BIN: -1101100010
=====
DEC: -224.0
HEX: -E0
```

```
BIN: -11100000
=====
DEC: -883.0
HEX: -373
BIN: -1101110011
=====
DEC: -338.0
HEX: -152
BIN: -101010010
=====
DEC: 229.0
HEX: E5
BIN: 11100101
=====
DEC: -902.0
HEX: -386
BIN: -1110000110
=====
DEC: 987.0
HEX: 3DB
BIN: 1111011011
=====
DEC: -735.0
HEX: -2DF
BIN: -1011011111
=====
DEC: -669.0
HEX: -29D
BIN: -1010011101
=====
DEC: -111.0
HEX: -6F
BIN: -1101111
=====
DEC: -687.0
HEX: -2AF
BIN: -1010101111
=====
DEC: -935.0
HEX: -3A7
BIN: -1110100111
=====
DEC: 922.0
HEX: 39A
BIN: 1110011010
=====
DEC: 882.0
HEX: 372
```

```
BIN: 1101110010
=====
DEC: 822.0
HEX: 336
BIN: 1100110110
=====
DEC: 808.0
HEX: 328
BIN: 1100101000
=====
DEC: 382.0
HEX: 17E
BIN: 101111110
=====
DEC: -391.0
HEX: -187
BIN: -110000111
=====
DEC: -763.0
HEX: -2FB
BIN: -1011111011
=====
DEC: 840.0
HEX: 348
BIN: 1101001000
=====
DEC: -877.0
HEX: -36D
BIN: -1101101101
=====
DEC: -721.0
HEX: -2D1
BIN: -1011010001
=====
DEC: -274.0
HEX: -112
BIN: -100010010
=====
DEC: -977.0
HEX: -3D1
BIN: -1111010001
=====
DEC: 28.0
HEX: 1C
BIN: 11100
=====
DEC: -521.0
HEX: -209
```

```
BIN: -1000001001
=====
DEC: 762.0
HEX: 2FA
BIN: 1011111010
=====
DEC: 8.0
HEX: 8
BIN: 1000
=====
DEC: -669.0
HEX: -29D
BIN: -1010011101
=====
DEC: -713.0
HEX: -2C9
BIN: -1011001001
=====
DEC: 240.0
HEX: F0
BIN: 11110000
=====
DEC: 212.0
HEX: D4
BIN: 11010100
=====
DEC: -5.0
HEX: -5
BIN: -101
=====
DEC: -930.0
HEX: -3A2
BIN: -1110100010
=====
DEC: -254.0
HEX: -FE
BIN: -11111110
=====
DEC: 39.0
HEX: 27
BIN: 100111
=====
DEC: 153.0
HEX: 99
BIN: 10011001
=====
DEC: 72.0
HEX: 48
```

```
BIN: 1001000
=====
DEC: -480.0
HEX: -1E0
BIN: -111100000
=====
DEC: 620.0
HEX: 26C
BIN: 1001101100
=====
DEC: -504.0
HEX: -1F8
BIN: -111111000
=====
DEC: -540.0
HEX: -21C
BIN: -1000011100
=====
DEC: 1231243.0
HEX: 12C98B
BIN: 100101100100110001011
=====
Time elapsed:0.0 seconds
```

Ejercicio 3 - Count Words

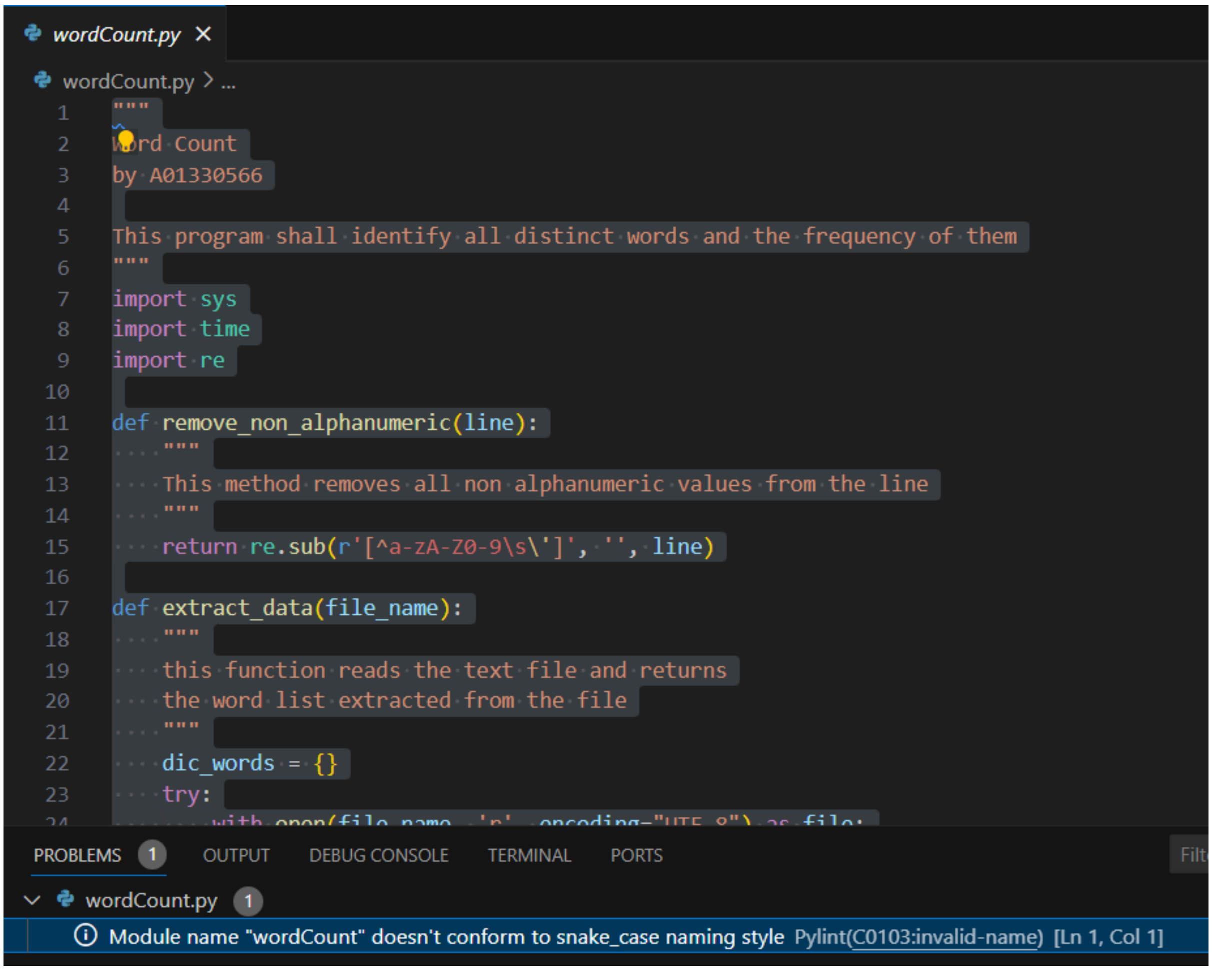
- Req1. The program shall be invoked from a command line. The program shall receive a file as parameter. The file will contain a words (presumable between spaces).
- Req 2. The program shall identify all distinct words and the frequency of them (how many times the word "X" appears in the file). The results shall be print on a screen and on a file named WordCountResults.txt.
  - All computation MUST be calculated using the basic algorithms, not functions or libraries.
- 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.
- Req 4. The name of the program shall be
  - wordCount.py
- Req 5. The minimum format to invoke the program shall be as follows:
  - python wordCount.py fileWithData.txt

- Req 6. The program shall manage files having from hundreds of items to thousands of items.
- 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.
- Req 8. Be compliant with PEP8.

Código Fuente

```python
"""
Word Count
by A01330566

This program shall identify all distinct words and the frequency of
them
"""
import sys
import time
import re

def remove_non_alphanumeric(line):
    """
    This method removes all non alphanumeric values from the line
    """
    return re.sub(r'[^a-zA-Z0-9\s\']', '', line)

def extract_data(file_name):
    """
    this function reads the text file and returns
    the word list extracted from the file
    """
    dic_words = {}
    try:
        with open(file_name, 'r', encoding="UTF-8") as file:
            for line in file:
                clean_line = remove_non_alphanumeric(line.strip())
                line_split = clean_line.strip().split()
                for word in line_split:
                    if word.lower() in dic_words:
                        dic_words[word.lower()] += 1
                    else:
                        dic_words[word.lower()] = 1
    except FileNotFoundError:
        print("File not found:", file_name)
        sys.exit(1)
    except UnicodeDecodeError:
        print("Error decoding file. Please ensure the file is UTF-8
encoded.")
        sys.exit(1)

    return dic_words

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python word_count.py InputFile.txt")
        sys.exit(1)

    start_time = time.time()
    filename = sys.argv[1]
    file_data = extract_data(filename)
    with open("WordCountResults.txt", 'w', encoding="UTF-8") as
results_file:
        for item in file_data.items():
            new_line = f"{item[0]} : {item[1]}"
            print(new_line)
            results_file.write(new_line +"\n")
        elapsed_time = time.time() - start_time
        print(f"Time elapsed:{elapsed_time} seconds")
        results_file.write(f"Time elapsed:{elapsed_time} seconds")
```

Resultados de PyLint



wordCount.py ×

wordCount.py > ...

```
"""
Word Count
by A01330566

This program shall identify all distinct words and the frequency of them
"""
import sys
import time
import re

def remove_non_alphanumeric(line):
    """
    This method removes all non alphanumeric values from the line
    """
    return re.sub(r'[^a-zA-Z0-9\s\']', '', line)

def extract_data(file_name):
    """
    this function reads the text file and returns
    the word list extracted from the file
    """
    dic_words = {}
    try:
```

```
Module name "wordCount" doesn't conform to snake_case naming style Pylint(C0103:invalid-name) [Ln 1, Col 1]
```

Ejemplo de Archivo Input

You got a fast car
And I want a ticket to anywhere
Maybe we make a deal
Maybe together we can get somewhere
Any place is better
Starting from zero, got nothing to lose
Maybe we'll make something
Me, myself, I got nothing to prove
You got a fast car
I got a plan to get us out of here
I've been working at the convenience store
Managed to save just a little bit of money
Won't have to drive too far
Just across the border and into the city
You and I can both get jobs
Finally, see what it means to be living

See, my old man's got a problem
He live with the bottle, that's the way it is
He says his body's too old for working
His body's too young to look like his
Mama went off and left him
She wanted more from life than he could give
I said, "Somebody's got to take care of him"
So, I quit school and that's what I did
You got a fast car
Is it fast enough so we can fly away?
Still gotta make a decision
Leave tonight, or live and die this way
So, I remember when we were driving, driving in your car
Speed so fast, I felt like I was drunk
City lights lay out before us
And your arm felt nice wrapped around my shoulder
And I, I, I had a feeling that I belonged
I, I, I had a feeling I could be someone, be someone, be someone
You got a fast car
We go cruising, entertain ourselves
You still ain't got a job
So I work in a market as a checkout girl
I know things will get better
You'll find work and I'll get promoted
We'll move out of the shelter
Buy a bigger house, live in the suburbs
So, I remember when we were driving, driving in your car
Speed so fast, I felt like I was drunk
City lights lay out before us
And your arm felt nice wrapped around my shoulder
And I, I, I had a feeling that I belonged
I, I, I had a feeling I could be someone, be someone, be someone
You got a fast car
I got a job that pays all our bills
You stay out drinking late at the bar
See more of your friends than you do of your kids
I'd always hoped for better
Thought maybe together you and me would find it
I got no plans, I ain't going nowhere
Take your fast car and keep on driving
So, I remember when we were driving, driving in your car
Speed so fast, I felt like I was drunk
City lights lay out before us
And your arm felt nice wrapped around my shoulder
And I, I, I had a feeling that I belonged
I, I, I had a feeling I could be someone, be someone, be someone
You got a fast car
Is it fast enough, so you can fly away?

You still gotta make a decision
Leave tonight, or live and die this way

Resultados del Ejemplo

```
PS C:\Users\rmuno\OneDrive\Documents\GitHub\A01330566_A4.2> python wordCount.py
TextFile.txt

you : 13
got : 14
a : 24
fast : 12
car : 10
and : 16
i : 45
want : 1
ticket : 1
to : 9
anywhere : 1
maybe : 4
we : 7
make : 4
deal : 1
together : 2
can : 4
get : 5
somewhere : 1
any : 1
place : 1
is : 4
better : 3
starting : 1
from : 2
zero : 1
nothing : 2
lose : 1
we'll : 2
something : 1
me : 2
myself : 1
prove : 1
plan : 1
us : 4
out : 6
of : 6
here : 1
i've : 1
```

```
been : 1
working : 2
at : 2
the : 8
convenience : 1
store : 1
managed : 1
save : 1
just : 2
little : 1
bit : 1
money : 1
won't : 1
have : 1
drive : 1
too : 3
far : 1
across : 1
border : 1
into : 1
city : 4
both : 1
jobs : 1
finally : 1
see : 3
what : 2
it : 5
means : 1
be : 10
living : 1
my : 4
old : 2
man's : 1
problem : 1
he : 3
live : 4
with : 1
bottle : 1
that's : 2
way : 3
says : 1
his : 3
body's : 2
for : 2
young : 1
look : 1
like : 4
mama : 1
```

```
went : 1
off : 1
left : 1
him : 2
she : 1
wanted : 1
more : 2
life : 1
than : 2
could : 4
give : 1
said : 1
somebody's : 1
take : 2
care : 1
so : 10
quit : 1
school : 1
did : 1
enough : 2
fly : 2
away : 2
still : 3
gotta : 2
decision : 2
leave : 2
tonight : 2
or : 2
die : 2
this : 2
remember : 3
when : 3
were : 3
driving : 7
in : 5
your : 9
speed : 3
felt : 6
was : 3
drunk : 3
lights : 3
lay : 3
before : 3
arm : 3
nice : 3
wrapped : 3
around : 3
shoulder : 3
```

```
had : 6
feeling : 6
that : 4
belonged : 3
someone : 9
go : 1
cruising : 1
entertain : 1
ourselves : 1
ain't : 2
job : 2
work : 2
market : 1
as : 1
checkout : 1
girl : 1
know : 1
things : 1
will : 1
you'll : 1
find : 2
i'll : 1
promoted : 1
move : 1
shelter : 1
buy : 1
bigger : 1
house : 1
suburbs : 1
pays : 1
all : 1
our : 1
bills : 1
stay : 1
drinking : 1
late : 1
bar : 1
friends : 1
do : 1
kids : 1
i'd : 1
always : 1
hoped : 1
thought : 1
would : 1
no : 1
plans : 1
going : 1
```

```
nowhere : 1
keep : 1
on : 1
Time elapsed:0.0 seconds
```