

2.- Apartado 2: Resolver ciertos problemas en Python.

Dado que a lo largo del año vamos a tener que trabajar bastante con Python, es necesario tener cierta base sobre los aspectos básicos del lenguaje. Para ello se propone la realización de los siguientes ejercicios que deberán ser subidos al repositorio GitHub del Apartado 1.

▪ Problema 1. Procesamiento de una lista de enteros.

Crea una función que reciba una lista de enteros por parámetro y devuelva otra lista, de acuerdo a las siguientes acciones:

1. Eliminar los números negativos de la lista.
2. Eliminar los valores que están repetidos, quedándonos con uno de ellos.
3. Ordena los números resultantes de menor a mayor.

Por ejemplo, si le pasara [4, -1, 2, 4, 3, -5, 2], debería retornar [2,3,4].

▪ Problema 2. Frecuencia de palabras en un texto.

Escribe una función que reciba por parámetro una lista de palabras y la ruta a un fichero de texto y devuelva un diccionario que muestre cuantas veces aparecen las distintas palabras de la lista en el fichero de texto. Haz un pequeño programa que la ponga a prueba.

Requisitos:

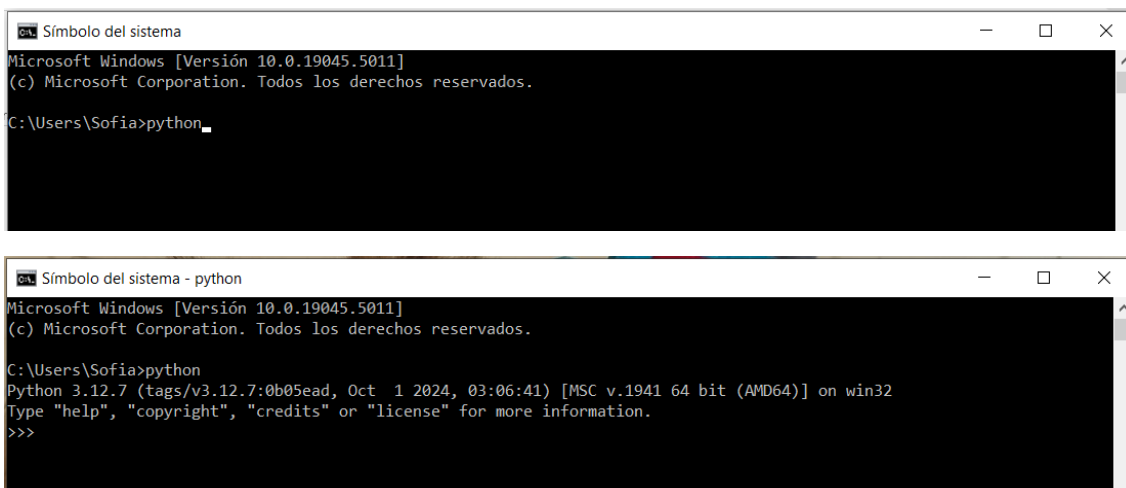
1. Eliminar signos de puntuación y convertir todo a minúsculas.
2. Usar un diccionario donde la clave sea la palabra y el valor su frecuencia.
3. Mostrar las palabras y sus frecuencias de forma ordenada por la palabra.

▪ Problema 3. Trabajo con conjuntos

Escribe una función que reciba dos listas de enteros y devuelva un diccionario con la siguiente información (**ES OBLIGATORIO USAR CONJUNTOS PARA CALCULARLOS**)

1. La intersección de ambos conjuntos (elementos comunes).
2. La unión de ambos conjuntos (todos los elementos sin duplicados).
3. La diferencia simétrica (elementos que están en uno u otro conjunto, pero no en ambos).

Abrimos Python:



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.5011]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Sofia>python

Símbolo del sistema - python
Microsoft Windows [Versión 10.0.19045.5011]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Sofia>python
Python 3.12.7 (tags/v3.12.7:0b05ead, Oct 1 2024, 03:06:41) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

2.1.- Problema 1. Procesamiento de una lista de enteros.

Crea una función que reciba una lista de enteros por parámetro y devuelva otra lista, de acuerdo a las siguientes acciones:

1. Eliminar los números negativos de la lista.
2. Eliminar los valores que están repetidos, quedándonos con uno de ellos.
3. Ordena los números resultantes de menor a mayor.

Por ejemplo, si le pasara [4, -1, 2, 4, 3, -5, 2], debería retornar [2,3,4].

CÓDIGO

```
# Lista inicial de números enteros.
listaInicial = [10, 8, -9, 3, 1, -3, 7, 8, 5, 13, 0, 9, 7, 3]
listaFinal = []
positivos = []
unicos = []

print("Lista de números enteros inicial:")
print(listaInicial)
print("-----")

def eliminar_negativos(lista):
    lista01 = []
    for elemento in lista:
        if elemento >= 0: # Solo añadimos los números no negativos
            lista01.append(elemento)

    # Mostramos la lista de los positivos obtenida.
    print("Lista de números positivos:")
    print(lista01)
    print("-----")
    return lista01

def eliminar_duplicados(lista):
    lista02 = []
    for numero in lista:
        if numero not in lista02:
            lista02.append(numero)

    # Mostramos la lista de los positivos obtenida.
    print("Lista de números sin duplicaciones:")
    print(lista02)
    print("-----")
    return lista02

# Llamamos a las dos primeras funciones:
positivos = eliminar_negativos(listaInicial)
unicos = eliminar_duplicados(positivos)

# Ordenamos la lista sin duplicados usando sorted() que no modifica la lista original
ordenada = sorted(unicos)
listaFinal = ordenada

# Mostramos la lista final que se solicita en el enunciado.
print("*****")
print("***** RESULTADO FINAL *****")
print("* Lista de números enteros positivos sin repeticiones y ordenados de menor a mayor: ")
print("* ", listaFinal)
print("*****")
```

El código como imagen desde el VSCode:

```
Tarea01_parte2_Problema01_SFM.py X
C: > Users > Usuario > Desktop > Tarea01PIA > Tarea01_parte2_Problema01_SFM.py > ...

1
2 # Lista inicial de números enteros.
3 listaInicial = [10, 8, -9, 3, 1, -3, 7, 8, 5, 13, 0, 9, 7, 3]
4 listaFinal = []
5 positivos = []
6 unicos = []
7
8 print("Lista de números enteros inicial:")
9 print(listaInicial)
10 print("-----")
11
12
13 def eliminar_negativos(lista):
14     lista01 = []
15     for elemento in lista:
16         if elemento >= 0: # Solo añadimos los números no negativos
17             lista01.append(elemento)
18
19     # Mostramos la lista de los positivos obtenida.
20     print("Lista de números positivos:")
21     print(lista01)
22     print("-----")
23     return lista01
24
25 def eliminar_duplicados(lista):
26     lista02 = []
27     for numero in lista:
28         if numero not in lista02:
29             lista02.append(numero)
30
31     # Mostramos la lista de los positivos obtenida.
32     print("Lista de números sin duplicaciones:")
33     print(lista02)
34     print("-----")
35     return lista02
36
37 # Llamamos a las dos primeras funciones:
38 positivos = eliminar_negativos(listaInicial)
39 unicos = eliminar_duplicados(positivos)
40
41 # Ordenamos la lista sin duplicados usando sorted() que no modifica la lista original
42 ordenada = sorted(unicos)
43 listaFinal = ordenada
44
45 # Mostramos la lista final que se solicita en el enunciado.
46 print("*****")
47 print("***** RESULTADO FINAL *****")
48 print("* Lista de números enteros positivos sin repeticiones y ordenados de menor a mayor: ")
49 print("* ", listaFinal)
50 print("*****")
51
```

Salida desde terminal de VSCode:

```
Tarea01_parte2_Problema01_SFM.py X
C: > Users > Usuario > Desktop > Tarea01PIA > Tarea01_parte2_Problema01_SFM.py > ...

1 |
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

PS C:\Users\Usuario> & C:/Users/Usuario/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Usuario/Desktop/Tarea01PIA/Tarea01_parte2_Problema01_SFM.py
Lista de números enteros inicial:
[10, 8, -9, 3, 1, -3, 7, 8, 5, 13, 0, 9, 7, 3]
-----
Lista de números positivos:
[10, 8, 3, 1, 7, 8, 5, 13, 0, 9, 7, 3]
-----
Lista de números sin duplicaciones:
[10, 8, 3, 1, 7, 5, 13, 0, 9]
-----
*****
***** RESULTADO FINAL *****
* Lista de números enteros positivos sin repeticiones y ordenados de menor a mayor:
* [0, 1, 3, 5, 7, 8, 9, 10, 13]
*****
PS C:\Users\Usuario>
```

2.2.- Problema 2. Frecuencia de palabras en un texto.

Escribe una función que reciba por parámetro una lista de palabras y la ruta a un fichero de texto y devuelva un diccionario que muestre cuantas veces aparecen las distintas palabras de la lista en el fichero de texto. Haz un pequeño programa que la ponga a prueba.

Requisitos:

1. Eliminar signos de puntuación y convertir todo a minúsculas.
2. Usar un diccionario donde la clave sea la palabra y el valor su frecuencia.
3. Mostrar las palabras y sus frecuencias de forma ordenada por la palabra.

CÓDIGO

```
import string

def contar_palabras(lista_palabras, ruta_fichero):

    # Inicializar diccionario con cero para cada palabra
    frecuencias = {palabra.lower(): 0 for palabra in lista_palabras}

    # Abrir y leer el fichero
    with open(ruta_fichero, 'r', encoding='utf-8') as fichero:
        for linea in fichero:
            # Eliminar puntuación y convertir a minúsculas
            linea = linea.translate(str.maketrans("", "", string.punctuation)).lower()
            palabras_linea = linea.split()

            # Contar las palabras
            for palabra in palabras_linea:
                if palabra in frecuencias:
                    frecuencias[palabra] += 1

    # Devolvemos el diccionario ordenado por palabra
    return dict(sorted(frecuencias.items()))

# --- Programa ---
if __name__ == "__main__":
    lista = ["cantar", "mayor", "un", "tiempo", "Señora", "jóvenes", "su", "Fortuna", "Elizabeth", "alba", "medianoche",
            "negro", "Canal", "y", "fantasma", "madre", "joven"]
    ruta = "C:/Users/Usuario/Desktop/Tarea01PIA/texto.txt" # Debe existir un fichero en la misma carpeta con el nombre
    indicado

    resultado = contar_palabras(lista, ruta)

    print("Frecuencia de palabras:")
    for palabra, freq in resultado.items():
        print(f"{palabra}: {freq}")
```

El código como imagen desde el VSCode:

```
Tarea01_parte2_Problema02_SFM.py X
C: > Users > Usuario > Desktop > Tarea01PIA > Tarea01_parte2_Problema02_SFM.py > ...
1
2 import string
3
4 def contar_palabras(lista_palabras, ruta_fichero):
5
6     # Inicializar diccionario con cero para cada palabra
7     frecuencias = {palabra.lower(): 0 for palabra in lista_palabras}
8
9     # Abrir y leer el fichero
10    with open(ruta_fichero, 'r', encoding='utf-8') as fichero:
11        for linea in fichero:
12            # Eliminar puntuación y convertir a minúsculas
13            linea = linea.translate(str.maketrans('', '', string.punctuation)).lower()
14            palabras_linea = linea.split()
15
16            # Contar las palabras
17            for palabra in palabras_linea:
18                if palabra in frecuencias:
19                    frecuencias[palabra] += 1
20
21    # Devolvemos el diccionario ordenado por palabra
22    return dict(sorted(frecuencias.items()))
23
24 # --- Programa ---
25 if __name__ == "__main__":
26     lista = ["cantar", "mayor", "un", "tiempo", "Señora", "jóvenes", "su", "Fortuna", "Elizabeth", "alba", "medianoche",
27             "negro", "Canal", "y", "fantasma", "madre", "joven"]
28     ruta = "C:/Users/Usuario/Desktop/Tarea01PIA/texto.txt" # Debe existir un fichero en la misma carpeta con el nombre indicado
29
30     resultado = contar_palabras(lista, ruta)
31
32     print("Frecuencia de palabras:")
33     for palabra, freq in resultado.items():
34         print(f"{palabra}: {freq}")
35
```

Salida desde terminal de VSCode:

```
PS C:\Users\Usuario> & C:/Users/Usuario/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Usuario/Desktop/Tarea01PIA/Tarea01_parte2_Problema02_SFM.py
Frecuencia de palabras:
alba: 0
canal: 0
cantar: 0
elizabeth: 0
fantasma: 0
fortuna: 2
joven: 1
jóvenes: 1
madre: 2
mayor: 1
medianoche: 0
negro: 0
señora: 2
su: 2
tiempo: 0
un: 5
y: 5
```

2.3.- Problema 3. Trabajo con conjuntos.

Escribe una función que reciba dos listas de enteros y devuelva un diccionario con la siguiente información **(ES OBLIGATORIO USAR CONJUNTOS PARA CALCULARLOS)**.

1. La intersección de ambos conjuntos (elementos comunes).
2. La unión de ambos conjuntos (todos los elementos sin duplicados).
3. La diferencia simétrica (elementos que están en uno u otro conjunto, pero no en ambos).

CÓDIGO

```
#Listas iniciales de números enteros
lista1 = [1, -2, 33, 14, -5, 0, 20, 3, -10, 22, 4, 2, 81, 102, 29]
lista2 = [4, 50, 2, 29, 16, 7, 81, 0, -2, 3, 33, -10, 20, 1, 93, 100, -34, -77]

def analizar_listas(lista1, lista2):

    # Convertir listas a conjuntos
    conjunto1 = set(lista1)
    conjunto2 = set(lista2)

    # Calcular intersección, unión y diferencia simétrica
    interseccion = conjunto1 & conjunto2      # o conjunto1.intersection(conjunto2)
    union = conjunto1 | conjunto2             # o conjunto1.union(conjunto2)
    diferencia_simetrica = conjunto1 ^ conjunto2 # o conjunto1.symmetric_difference(conjunto2)

    # Devolver resultados en un diccionario
    resultado1 = {
        "interseccion": interseccion,
        "union": union,
        "diferencia_simetrica": diferencia_simetrica
    }

    return resultado1

resultado1 = analizar_listas(lista1, lista2)
print(resultado1)
```

El código como imagen desde el VSCode:

```
Tarea01_parte2_Problema03_SFM.py X
C: > Users > Usuario > Desktop > Tarea01PIA > Tarea01_parte2_Problema03_SFM.py > ...
1
2     #Listas iniciales de números enteros
3 lista1 = [1, -2, 33, 14, -5, 0, 20, 3, -10, 22, 4, 2, 81, 102, 29]
4 lista2 = [4, 50, 2, 29, 16, 7, 81, 0, -2, 3, 33, -10, 20, 1, 93, 100, -34, -77]
5
6 def analizar_listas(lista1, lista2):
7
8     # Convertir listas a conjuntos
9     conjunto1 = set(lista1)
10    conjunto2 = set(lista2)
11
12    # Calcular intersección, unión y diferencia simétrica
13    interseccion = conjunto1 & conjunto2          # o conjunto1.intersection(conjunto2)
14    union = conjunto1 | conjunto2                 # o conjunto1.union(conjunto2)
15    diferencia_simetrica = conjunto1 ^ conjunto2  # o conjunto1.symmetric_difference(conjunto2)
16
17    # Devolver resultados en un diccionario
18    resultado1 = {
19        "interseccion": interseccion,
20        "union": union,
21        "diferencia_simetrica": diferencia_simetrica
22    }
23
24    return resultado1
25
26 resultado1 = analizar_listas(lista1, lista2)
27 print(resultado1)
28
```

Salida desde terminal de VSCode:

```
PS C:\Users\Usuario> & C:/Users/Usuario/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Usuario/Desktop/Tarea01PIA/Tarea01_parte2_Problema03_SFM.py
{'interseccion': {0, 1, 33, 3, 4, 2, 81, 20, -10, 29, -2}, 'union': {0, 1, 2, 3, 4, 7, 14, 16, 20, 22, 29, 33, 50, -77, 81, 93, -34, 100, 102, -10, -5, -2},
'diferencia_simetrica': {7, 14, 16, 22, 93, -34, 100, 102, 50, -77, -5}}
PS C:\Users\Usuario>
```