Informe

**1. Ordenar nombres**

A) Para este ejercicio, comencé abriendo el archivo nombres.txt que se nos brindó en la consigna. Utilicé la sentencia with open() para abrir el archivo en modo lectura ('r'), dado que no vamos a modificarlo. Para la codificación, empleé utf-8, ya que es la más común para manejar diferentes tipos de caracteres, y evita problemas con caracteres especiales o acentos, a diferencia de la codificación predeterminada de Python que no es compatible con algunos archivos

Ej en el código:

with open('nombres.txt', 'r', encoding='utf-8') as archivo:

B) Luego, implementé un control de errores para verificar si el archivo está vacío. Utilicé el método readlines() para leer todo el contenido del archivo y convertirlo en una lista. Si la lista está vacía, el programa lanza un ValueError con el mensaje "El archivo está vacío" utilizando raise. Esto detiene la ejecución y muestra una excepción que se captura más adelante en el bloque except

Ej en el código:

contenido = archivo.readlines()

if not contenido:

raise ValueError("El archivo está vacío.")

C) Para procesar las líneas del archivo, utilicé un bucle for que recorre cada línea de la lista generada por readlines(). Para cada línea, usé el método .strip() para eliminar los saltos de línea \n y los espacios vacíos. Luego, las agregué a una nueva lista usando .append().

Ej lógico:

Archivo:

Primera línea

Segunda línea

Tercera línea

Lista generada con readlines:

['Primera línea\n', 'Segunda línea\n', 'Tercera línea\n']

Nueva lista con strip:

['Primera línea', 'Segunda línea', 'Tercera línea']

D) Para ordenar los nombres de manera adecuada (sin tener en cuenta mayúsculas o acentos), implementé la función quitar\_acento(). En esta función, utilicé un diccionario que mapea letras acentuadas a sus versiones sin acento, por ejemplo, 'á': 'a'. Luego, un bucle for recorre el diccionario y reemplaza las letras acentuadas en cada palabra También convertí todo a minúsculas cuando se retorna la palabra con .lower() para evitar que las mayúsculas afecten el orden

Ej en codigo:

def quitar\_acento(palabra):

acentos = {

'á': 'a', 'é': 'e', 'í': 'i', 'ó': 'o', 'ú': 'u',

'Á': 'A', 'É': 'E', 'Í': 'I', 'Ó': 'O', 'Ú': 'U'

}

for acento, sin\_acento in acentos.items():

palabra = palabra.replace(acento, sin\_acento)

return palabra.lower()

E) Finalmente, para completar el proceso de ordenación, creé una lista llamada lista\_limpia donde almacené los nombres sin acentos y en minúsculas. Luego utilicé sorted() para ordenar esta lista alfabéticamente

lista\_limpia = []

for palabra in listaNombresOrdenados:

lista\_limpia.append(quitar\_acento(palabra))

F) El manejo de errores es fundamental en este tipo de programas para evitar que situaciones inesperadas (como la ausencia del archivo o contenido no válido) generen fallos en el código. Utilicé un bloque try-except con varias excepciones específicas:

* FileNotFoundError: Si el archivo no existe, se captura esta excepción y se muestra un mensaje adecuado.
* ValueError: Si el archivo está vacío, como se explicó anteriormente, se captura esta excepción.
* Exception: Para cualquier otro error que no se haya previsto, se captura esta excepción general.

Ej en código:

except FileNotFoundError:

print("Error: El archivo no existe")

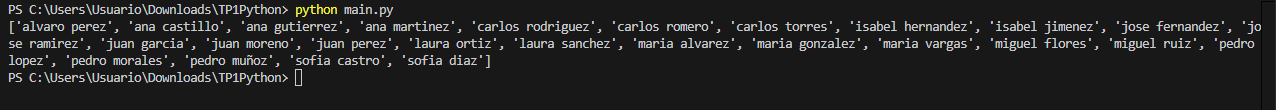
except ValueError as ve:

print(f"Error: {ve}")

except Exception as e:

print(f"Error: {e}")

Al agregar un print(lista\_limpia) y ejecutar la función:

ordernarNombres()

**2. Convertir nombres a diccionario**

A) Para este ejercicio, comencé con la función removeApellidos(lista), que recibe una lista de nombres completos y devuelve una lista con solo los nombres, eliminando los apellidos. Para lograr esto, utilicé el método .split() para separar los nombres completos por espacios. Luego, extraje el primer nombre [0] y lo agregué a una nueva lista llamada listaSinApellido.

Ej en código:

def removeApellidos(lista):

listaSinApellido = []

for nombreCompleto in lista:

nombre = nombreCompleto.split()[0]

listaSinApellido.append(nombre)

return listaSinApellido

B) Luego, la función principal convertirDiccionario(listaNombres) convierte la lista de nombres (ya sin apellidos) en un diccionario donde las claves son los nombres y los valores representan la frecuencia de repetición de cada nombre

Para empezar, agregué un control de errores que verifica si la lista está vacía (el mismo que el ejercicio 1)

C) A continuación, utilicé un bucle for que recorre la lista de nombres y construye el diccionario. Para cada nombre en la lista, reviso si ya existe en el diccionario:

* Si el nombre ya está presente, incrementó su contador
* Si no está, lo agrego al diccionario con un valor de 1

Ej en código:

for nombres in listaNombres:

if nombres in diccionario:

diccionario[nombres] += 1

else:

diccionario[nombres] = 1

D) Finalmente, use el manejo de errores con varios bloques except para capturar diferentes tipos de excepciones (el mismo que el ejercicio 1)

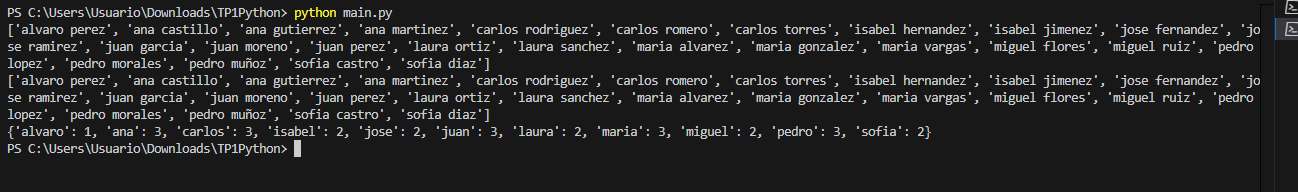
E)La desventaja de esta función es que debe recibir la lista ordenada, de modo que:

lista = ordernarNombres()

dicNombresFrecuencia = convertirDiccionario(lista)

En la consola, se imprimirá 2 veces la lista ordenada, situación que no corregí porque se

resuelve más adelante con la implementación de Tkitner

En consola:

**3. Generar archivo de frecuencia de nombres**

A) Para este ejercicio, el objetivo fue tomar el diccionario de nombres generado en el ejercicio anterior y escribir su contenido en un archivo de texto. La función txtFrecuenciaNombres(dicNombres) toma el diccionario de nombres y guarda la frecuencia de cada nombre en un archivo llamado frecuencia\_nombres.txt

B) Para escribir en el archivo, utilicé un bloque with open() que asegura que el archivo se abra en modo de escritura ('w'). Si el archivo no existe, se creará automáticamente. El método with también se encarga de cerrar el archivo una vez que se termina de escribir, lo que evita errores.

Ej en código:

with open('frecuencia\_nombres.txt', 'w') as archivo:

C) En el bucle for, recorremos el diccionario dicNombres, donde nombre es la clave (el nombre de la persona) y cantidad es el valor (la cantidad de veces que aparece el nombre). Dentro del bucle, utilizamos archivo.write() para escribir cada par nombre-cantidad en una nueva línea del archivo de texto

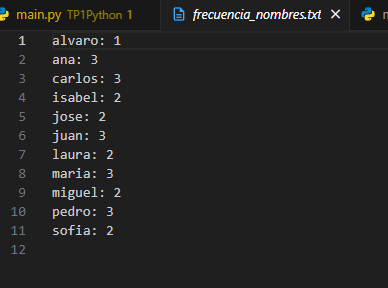
Ej en código:

for nombre, cantidad in dicNombres.items():

archivo.write(f"{nombre}: {cantidad} \n")

Al ejecutar la función:

txtFrecuenciaNombres(dicNombresFrecuencia)



**4. Generar gráfico de frecuencia de nombre**

A) En este ejercicio, la función graficoFrecuenciaNombres() se encarga de visualizar la frecuencia de los nombres a partir de los datos contenidos en el archivo nombres.txt Utilizo la biblioteca matplotlib para crear un gráfico de barras que representa cuántas veces aparece cada nombre en la lista

B) Para comenzar, la función abre el archivo nombres.txt, de la misma forma qué ejercicios anteriores

C) A continuación, llamo a la función removeApellidos() para extraer solo los nombres, y utilizo Counter del módulo collections para contar la frecuencia de cada nombre en la lista. Esto proporciona un objeto Counter que se comporta como un diccionario, donde las claves son los nombres y los valores son las frecuencias

Ej en código:

listaNombres = removeApellidos(listaNombreCompleto)

contadorFrecuencia = Counter(listaNombres)

D) Luego, utilizo un bucle for para contar el número total de nombres en listaNombres, que se almacena en la variable contadorNombres. Esto sirve para definir cuántos nombres se mostrarán en el gráfico

Ej en código:

contadorNombres = 0

for nombres in listaNombres:

contadorNombres += 1

E) Creó un diccionario con dict que contiene los nombres más comunes y su frecuencia, utilizando el método most\_common() de Counter

nombres\_comunes = dict(contadorFrecuencia.most\_common(contadorNombres))

F) Para visualizar los datos, se crea un gráfico de barras con plt.bar(). Se configura el tamaño de la figura y se añaden etiquetas y un título al gráfico. Para mejorar la presentación, los nombres en el eje x se rotan 45 grados para una mejor legibilidad.

Ej en código:

plt.figure(figsize=(12, 6))

bars = plt.bar(nombres\_comunes.keys(),nombres\_comunes.values())

plt.title("Los nombres mas frecuentes")

plt.xlabel("Nombres")

plt.ylabel("Frecuencia")

plt.xticks(rotation=45, ha='right')

G) Para añadir mayor claridad al gráfico, utilizo un bucle for para mostrar la frecuencia de cada nombre sobre las barras correspondientes, lo que permite al lector visualizar la cantidad exacta de veces que cada nombre aparece.

Ej en código:

for bar in bars:

height = bar.get\_height()

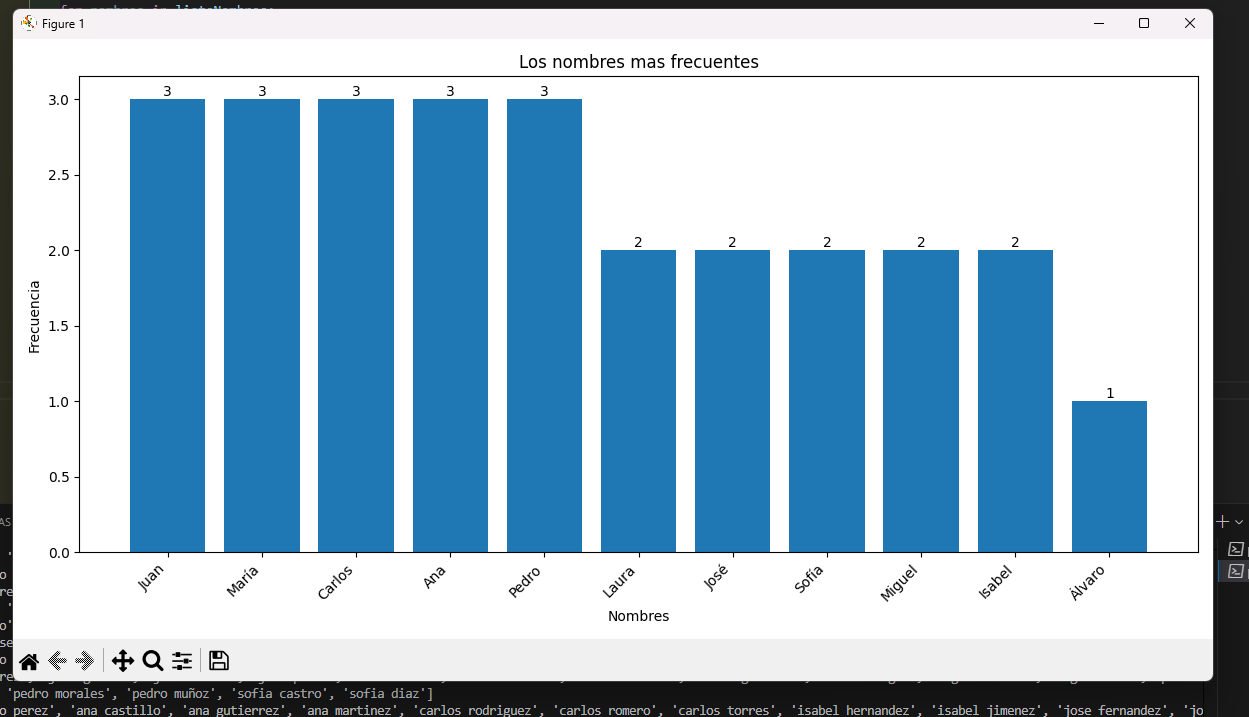
plt.text(bar.get\_x() + bar.get\_width() / 2., height, f"{height}", ha="center", va="bottom")

H) Luego, agregué la línea plt.savefig('grafico\_frecuencia\_nombres.png'), que guarda el gráfico generado como un archivo de imagen PNG.

Ej en código:

plt.savefig('grafico\_frecuencia\_nombres.png')

I) Finalmente, utilizo plt.tight\_layout() para ajustar automáticamente los márgenes y plt.show() para mostrar el gráfico en pantalla.



**5. Inicial en mayúscula de palabras con excepciones**

A) La función inicialMayuscula(cadena) tiene como objetivo convertir en mayúscula la primera letra de cada palabra en una cadena, exceptuando ciertas palabras como "y", "el", "la", "en", "de", "con", "si", "es", "o", "ser"

B) La función comienza definiendo una lista de excepciones, listaExepciones, que contiene las palabras que no deben ser transformadas. Esto asegura que solo se apliquen cambios a las palabras que no están en esta lista

listaExepciones = ["y", "el", "la", "en", "de"]

C) Luego, se divide la cadena de entrada en una lista de palabras utilizando el método split(). Esto permite iterar sobre cada palabra de forma individual.

palabras = cadena.split()

D) A continuación, se utiliza un bucle for para recorrer cada palabra en la lista. Para cada palabra, se verifica si no está en la lista de excepciones. Si la palabra no es una excepción, se convierte su primera letra y se reconstruye la cadena original utilizando el método replace() debido a que una cadena en python no es modificable

for palabra in palabras:

if palabra not in listaExepciones:

cadena = cadena.replace(palabra, palabra[0].upper() + palabra[1:])

Para imprimir en consola:

cadena = "hola, como estas y holah hh?"

inicialMayuscula(cadena)

**6. Implementación de la Interfaz Gráfica con Tkinter**

A) El código principal comienza importando Tkinter y las funciones previamente definidas desde un módulo llamado funciones. También se definen variables globales lista\_nombres y diccionario\_nombres para almacenar los resultados de las operaciones realizadas en la aplicación

B)Instanciamos Tk con una ventana principal a la que llamamos root, luego definimos el título y el tamaño de la interfaz gráfica

Ej en codigo:

root = tk.Tk()

root.title("Python, parcial n°1")

root.geometry("300x300")

C)Para mostrar las funciones anteriores, genere funciones para pasarlas de comandos a un botón en específico, también implemente manejo de errores con messagebox tales como:

-No se puede crear el diccionario sin ordenar la lista

-Mostrar error si el archivo no existe

-No se puede generar el txt de frecuencias si no se creo el diccionario

-Y errores no contemplados pero posibles con except Exception as e: