Sprint 7 - Juan Romero R.

Antes de comenzar quiero aclarar que he decidido siempre que me ha sido posible, crear funciones para cada parte del codigo, para hacer una buena práctica de hacer códigos estructurados ,fácilmente entendibles y manipulables. Si bien es cierto que hay ciertos trozos de código que son tan sencillos que no haría falta, pero por practicar los he mantenido.

Nivell 1

- Exercici 1

Calculadora de l'índex de massa corporal

Escriu una funció que calculi l'IMC ingressat per l'usuari/ària, és a dir, qui ho executi haurà d'ingressar aquestes dades.

```
def imc_calculo(peso,altura) :
    categorias_de_peso = {
        "Bajo peso": (1, 18.5),
        "Peso normal": (18.5, 24.9),
        "Sobrepeso": (25, 29.9),
        "Obesidad": (30, 999)
    }

    IMC= peso / (altura ** 2)
    imc_redondeado = f"{IMC:.2f}"

    for categoria, (minimo, maximo) in categorias_de_peso.items():
     if minimo<= IMC <= maximo:
         return imc_redondeado, categoria
    return imc_redondeado, "Fuera de rango"

peso=float(input("Introduce tu peso en KG"))
altura=float(input("Introduce tu altura en Metros"))

imc_redondeado, resultado = imc_calculo(peso, altura)
print("Tu IMC es:", imc_redondeado, "Tu categoria de IMC esta dentro
de:", resultado)
```

En mi caso he decidido escribir el código pensando que aunque sea un cálculo sencillo, que el código sea lo más entendible que sea

posible (hasta donde se) y que sea fácil de cambiar en caso de que sea necesario.

Para ello he decidido crear una función que se encargar de gestionar y clasificar el resultado de la operación de IMC (se encarga tanto de decir como se debe de ejecutar la operación a como se debe clasificar):

```
def imc_calculo(peso,altura) :
    categorias_de_peso = {
        "Bajo peso": (1, 18.5),
        "Peso normal": (18.5, 24.9),
        "Sobrepeso": (25, 29.9),
        "Obesidad": (30, 999)
    }

    IMC= peso / (altura ** 2)
    imc_redondeado = f"{IMC:.2f}"

    for categoria, (minimo, maximo) in categorias_de_peso.items():
     if minimo<= IMC <= maximo:
         return imc_redondeado, categoria
    return imc_redondeado, "Fuera de rango"
```

La creamos con el nombre “imc_calculo” y espera tanto un valor de altura y otro valor de peso.:

```
def imc_calculo(peso,altura) :
```

Luego definimos un diccionario con las categorías de como se clasifican los valores del índice de masa corporal, así podemos editarlo fácilmente ya que es bastante accesible por si en algún momento esta clasificación cambia.

```
    categorias_de_peso = {
        "Bajo peso": (1, 18.5),
        "Peso normal": (18.5, 24.9),
        "Sobrepeso": (25, 29.9),
        "Obesidad": (30, 999)
    }
```

Después definimos la operación para calcular el “IMC” en este caso la operación que realizamos es peso entre (altura elevado a dos)

```
IMC= peso / (altura ** 2)
```

Y luego realizando pruebas me di cuenta que si directamente imprimimos este valor nos daba una cantidad muy grande de decimales. Así que usamos un redondeo del resultado:

```
imc_redondeado = f"{IMC:.2f}"
```

Ahora empezamos con un bucle para buscar y clasificar el resultado según nuestro diccionario de categorías:

```
    for categoria, (minimo, maximo) in categorias_de_peso.items():
     if minimo<= IMC <= maximo:
         return imc_redondeado, categoria
    return imc_redondeado, "Fuera de rango"
```

Aquí vemos como para cada categoría dentro de nuestro diccionario se compuerba si el “imc” es mayor que el mínimo y menor que el máximo (ambos incluidos) nos retornara el imc redondeado junto con la categorioa, en caso de no encontrar ninguno, nos retornara el imc redondeado y un mensaje de fuera de rango

```
peso=float(input("Introduce tu peso en KG"))
altura=float(input("Introduce tu altura en Metros"))

imc_redondeado, resultado = imc_calculo(peso, altura)
print("Tu IMC es:", imc_redondeado, "Tu categoria de IMC esta dentro
de:", resultado)
```

Ahora, estableceremos dos variables que serán inputs de nuestro peso y altura, especificamos la magnitud para que el usuario sea conciente del dato a introducir. También es importante ya que son datos con comas, castear estos datos a floats que es el tipo de dato que se espera en nuestro programa.

Una vez hecho esto establecemos dos valores, el imc_redondeado y el resultado que esto valores nos lo retornara la función “imc:calculo” que hemos creado previamente.

Una vez ejecutado todo imprimimos el resultado.

- Exercici 2

Convertidor de temperatures.

Existeixen diverses unitats de temperatura utilitzades en diferents contextos i regions. Les més comunes són Celsius (°C), Fahrenheit (°F) i Kelvin (K). També existeixen altres unitats com Rankine (°Ra) i Réaumur (°Re). Selecciona almenys 2 i construeix el convertidor.

Cabe destacar que este ejercicio ya lo había hecho con anterioridad.

Desde un principio puede ser un Código bastante sencillo de escribir si solo nos centramos en dos magnitudes. Aun así he decidido coger 3 de las 5 magnitudes para escribir el código.

También añado que sería “muy fácil” escribirlo con un par de inputs, 4 en realidad, y varios if. Sin embargo, en este ejercicio realizamos en su momento una práctica para que el código sea mantenible y entendible en el tiempo, dando la oportunidad (que bien este no es un caso donde las operaciones cambiarían con el tiempo) de poder ser fácilmente modificable.

También como apunte he usado números para escoger la magnitud de la temperatura sin embargo quizás sea más claro usar método upper en los inputs para tratar directamente con el nombre de la magnitud.

En primera instancia, creamos un diccionario asignando un valor a cada unidad de temperatura. Esto nos ayudara a poder introducir nuevas magnitudes de temperatura fácilmente sin tener que editar gran parte del codigo

```
unidades_de_temperatura = {
    1: "Celsius",
    2: "Fahrenheit",
    3: "Kelvin"
}
```

Luego creamos una función para cada conversión, esta parte la he hecho así para, pese a que este no es el caso, poder editar fácilmente las operaciones en caso de variar con el tiempo. En caso de añadir una nueva magnitud en el diccionario, a su vez será más fácil añadir una nueva función para su conversión.

```
##funcion para conversion desde Celcius
def celsius_fahrenheit(temp):
    return (temp * 1.8) + 32

def celsius_kelvin(temp):
    return temp + 273.15

##Funcion para conversion desde Farenheit
def fahrenheit_celsius(temp):
    return (temp - 32) / 1.8

def fahrenheit_kelvin(temp):
    return ((temp - 32) / 1.8) + 273.15

##Fncion para conversion desde Kelvin

def kelvin_celsius(temp):
    return temp - 273.15

def kelvin_fahrenheit(temp):
    return ((temp - 273.15) * 1.8) + 32
```

Una vez hecho esta parte, debemos indicar la relación entre las magnitudes que el usuario escoge, para esto creamos un diccionario, donde la llave de la temperatura de origen y de destino nos dará qué tipo de función para la conversión se usará.

```
conversiones = {
    (1, 2): celsius_fahrenheit,
    (1, 3): celsius_kelvin,
    (2, 1): fahrenheit_celsius,
    (2, 3): fahrenheit_kelvin,
    (3, 1): kelvin_celsius,
    (3, 2): kelvin_fahrenheit
}
```

Luego creamos la función para convertir la temperatura, esta esperará una temperatura de origen, una unidad de origen y una unidad de destino. Luego nos decimos que si la secuencia de unidad de origen y la de destino está dentro del diccionario de conversiones nos retorne la conversión y la temperatura de origen, sino nos de un mensaje de conversión no soportada,

esto evitará que el programa de error al seleccionar conversiones erróneas en este caso entre iguales

```
def convertir_temperatura(temp_origen, unidad_origen, unidad_destino):
    if (unidad_origen, unidad_destino) in conversiones:
        return conversiones[(unidad_origen,
unidad_destino)](temp_origen)
    else:
        return "Conversión no soportada"
```

Ahora viene donde definimos el input para selecciona unidad donde espera un mensaje(mensaje correspondiente a temperatura de origen o destino). Para ellos En el input lo casteamos a INT y le damos una instrucción al usuario donde le asignamos el numero correspondiente a cada temperatura, este numero es el que nos dirá que magnitud vamos a tratar. En caso de introducir otro número corresponda saltara un mensaje de error para que introduzca un valor válido.

```
def seleccionar_unidad(mensaje):
    while True:
        try:
            unidad = int(input(f"{mensaje}"
                               "\n 1 - Celsius"
                               "\n 2 - Fahrenheit"
                               "\n 3 - Kelvin"
                               "\n Número de la unidad: "))
            if unidad in unidades_de_temperatura:
                return unidad
            else:
                print("Por favor, introduce un número válido entre 1 y
3.")
        except ValueError:
            print("Por favor, introduce un número entero.")
```

Aunque la siguiente función no es estrictamente necesaria crearla, por continuar con la practica la crearemos, es simplemente para solicitar el valor de la temperatura convirtiendo el valor a float.

```
def solicitar_temperatura(solicitar):
    return float(input(f"{solicitar}"))
```

Ahora usaremos todas las funciones correspondientes para solicitar los datos al usuario.

Primeramente, la variable unidad_origen donde le asignamos la función seleccionar_unidad previamente creada con un mensaje personalizado para indicar que es la unidad de origen.

Creamos la variable temp_origen donde usamos la función solicitar_temperatura con el mensaje correspondiente.

Y por ultimo la variable para la unidad_destino que sigue la misma lógica que la unidad_origen

```
unidad_origen = seleccionar_unidad("Introduce el número correspondiente
a la unidad de temperatura de origen")
temp_origen = solicitar_temperatura("Introduce la temperatura:")
unidad_destino = seleccionar_unidad("Introduce el número
correspondiente a la unidad de temperatura de destino")
```

Ahora que tenemos todos los datos que necesitamos para usar la conversión, la ejecutamos. Cabe decir que tanto nombre_unidad_origen como nombre_unidad_destino fueron creados para manejar más fácilmente el menaje final del resultado y llamar correctamente las magnitudes dentro del diccionario.

Luego la variable resultado es donde nos dará el valor de la función convertir_temperatura que ejecutara la operación necesaria.

```
nombre_unidad_origen = unidades_de_temperatura[unidad_origen]
nombre_unidad_destino = unidades_de_temperatura[unidad_destino]
resultado = convertir_temperatura(temp_origen, unidad_origen,
unidad_destino)
```

Una vez hecho esto imprimimos el resultado.

```
print(f"{temp_origen}º {nombre_unidad_origen} = {resultado:.2f}º
{nombre_unidad_destino}")
```

- Exercici 3

Comptador de paraules d'un text.

Escriu una funció que donat un text, mostri les vegades que apareix cada paraula.

Primeramente, importamos counter y string para manejar el los strings y contar fácilmente

```
from collections import Counter
import string
```

Con tal de manejar los apostrofes (pensando en el catalán) y tratarlos como una palabra diferente creamos una variable donde usaremos el método punctuation indicándole además que reemplace los apostrofes (‘) por espacios.

```
punctuation_apost = string.punctuation.replace("'", "")
```

Luego en la variable le asignamos un input que será el texto que el usuario introducirá para contar las palabras. Luego usamos sobre el texto “lower” para ponerlo todo en minúscula.

```
texto = input("Introduce el texto para contar la frecuencia de las
palabras: ")
texto = texto.lower()
```

Ahora crearemos una “tabla” donde convertiremos el texto y limpiaremos los símbolos de puntuación comunes con la variable punctuation_apost una vez hecho esto convertiremos de vuelta el texto sin símbolos a nuestro formato con .translate

```
tabla = str.maketrans('', '', punctuation_apost)
texto_limpio = texto.translate(tabla)
```

Ahora con el método Split separaremos cada palabra del texto.

```
palabras = texto_limpio.split()
```

Ahora creamos la variables contador donde usaremos Counter de la librería que importamos antes para crear un diccionario donde cada palabra es una llave y el valor será la cantidad de palabras que tiene la llave

El total lo he añadido para contar el total de palabras que hay en el texto

```
contador = Counter(palabras)
total = len(palabras)
```

Ahora printaremos los resultados donde primero mostramos el total de las palabras y luego a través de un bucle for se creará una linea para cada palabra y valor del contador previamente creado, ademoas con sorted lo ordenamos por orden alfabético.

```
print(f" Hay un total de {total} palabras distribuidas de la siguiente
forma:")
print("Frecuencia de cada palabra ordenada alfabéticamente:")
for palabra in sorted(contador):
    frecuencia = contador[palabra]
    print(f"{palabra}: {frecuencia}")
```

- Exercici 4

Diccionari invers.

Resulta que el client té una enquesta molt antiga que s'emmagatzema en un diccionari i els resultats els necessita al revés, és a dir, intercanviats les claus i els valors. Els valors i claus en el diccionari original són únics; si aquest no és el cas, la funció hauria d'imprimir un missatge d'advertiment.

Establecemos un diccionario para simular la encuesta del cliente, asignamos clave y valores aleatorios, en este caso intencionadamente hemos puesto un valor repetido para simular el error

```
encuesta = {
    "Esto": 1,
    "Es": 2,
    "Una": 3,
    "Prueba": 4,
    "loco": 4
}
```

Luego asignamos a una variable el diccionario con las claves inversas, este está vacío pues se añadirán aquí posteriormente.

```
encuesta_invertida = {}
```

Ahora en un bucle iterando entre los valores y las llaves de los elementos del diccionario “encuesta” (usamos .items que es el método que indica que son los elementos dentro del diccionario)

Primeramente, en una condición le indicamos que si el valor ya está en encuesta invertida, nos dé el mensaje de error.

Luego simplemente indicamos que en la encuesta invertida los valores son las llaves.

```
for k, v in encuesta.items():
    if v in encuesta_invertida:
        print(f"Valor duplicado encontrado para la clave '{v}' con las
entradas '{encuesta_invertida[v]}' y '{k}'")
        continue
    encuesta_invertida[v] = k
```

Nivell 2

Exercici 1

Diccionari invers amb duplicats

Continuant amb l'exercici 4 del nivell 1: al client es va oblidar de comentar un detall i resulta que els valors en el diccionari original poden duplicar-se i més, per la qual cosa les claus intercanviades poden tenir duplicats. En tals casos, els valors del diccionari resultant hauran d'emmagatzemar-se com una llista. Tingues en compte que un valor únic no ha de ser una llista.

Al igual que el ejercicio anterior establecemos un diccionario con la encuesta, también establecemos dos llaves con el mismo valor.

```
encuesta = {
    "Esto": 1,
    "Es": 2,
    "Una": 3,
    "Prueba": "ane",
    "Otra": "ane"

}
```

Establecemos una variable asignándole un diccionario vacío donde se guardará los nuevos valores invertidos.

```
encuesta_invertida = {}
```

Ahora iniciamos un bucle para invertir los valores de los objetos de la encueta. Además indicamos que si ya existe un repetido, este se añada en una lista sobre la nueva llave. (en el primer else)

```
for k, v in encuesta.items():
    if v not in encuesta_invertida:
        encuesta_invertida[v] = k
    else:
        if isinstance(encuesta_invertida[v], list):
            encuesta_invertida[v].append(k)
        else:
            encuesta_invertida[v] = [encuesta_invertida[v], k]
```

Por ultimo imprimimos el resultado.

```
print(encuesta_invertida)
```

Exercici 2

Conversió de tipus de dades

El client rep una llista de dades i necessita generar dues llistes, la primera on estaran tots els elements que es van poder convertir en flotants i l'altra on estan els elements que no es van poder convertir. Exemple de la llista que rep el client: ['1.3', 'one' , '1e10' , 'seven', '3-1/2', ('2',1,1.4,'not-a-number'), [1,2,'3','3.4']]

Primero creamos una función y definimos las listas tras aplicar la conversión a float. Una para el conjunto y una para cada situación según si se puede convertir en float o no

```
def procesar_lista(lista_prueba):
    resultado = []
    lista_float = []
    lista_nofloat = []
```

Ahora definimos una función para desanidar los elementos de la lista y comprobar si son floats, esto quiere decir si un elemento de la lista es una tupla, o lista esta función se encargará de sacarlos y ponerlo en la lista base tras eso comprobara si los elementos se pueden convertir a float (num = float(elemento)). Si el elemento no se puede convertir a flotante lo gestionaremos con un try donde definiremos en ValueError de except que los que no se puedan convertir porque el casteo falle se añadan a la lista de no floats

```
    def aplanar_y_convertir(elemento):
        if isinstance(elemento, (list, tuple)):
            for item in elemento:
                aplanar_y_convertir(item)
        else:
            resultado.append(elemento)
            try:
                num = float(elemento)
                lista_float.append(elemento)
            except ValueError:
                lista_nofloat.append(elemento)
```

Aquí es donde indicamos que todo lo anterior se ejecutara para cada elemento de la lista y nos devolverá las lista que hemos creado para cada una de las situaciones (Aunque la lista combinada no se requiere, no esta demás tenerla para comprobar como ha evaluado el programa toda la lista, y si esta todo correcto)

```
    for i in lista_prueba:
        aplanar_y_convertir(i)
    return resultado, lista_float, lista_nofloat
```

Ahora simplemente definimos la lista sobre la que queremos trabajar e ejecutamos el programa

```
lista = [ '1.3', 'one' , '1e10' , 'seven', '3-1/2', ('2',1,1.4,['not-a-number']), [1,2,'3','3.4']]

procesar_lista(lista)
print(lista_float)
print(lista_nofloat)
```

Nivell 3

Exercici 1

Comptador i endreçador de paraules d'un text.

El client va quedar content amb el comptador de paraules, però ara vol llegir arxius TXT i que calculi la freqüència de cada paraula ordenades dins de les entrades habituals del diccionari segons la lletra amb la qual comencen, és a dir, les claus han d'anar de la A a la Z i dins de la A hem d'anar de la A la Z. Per exemple, per a l'arxiu "tu_me_quieres_blanca.txt" la sortida esperada seria:

Primeramente, importamos las librerías necesarias. Además, como volvemos a trabajar con punctuation de string en esta ocasión añadiremos a este metodo los símbolos de puntuación que existen en español y no en inglés.

```
from collections import Counter
import string
puntuacion_extra = string.punctuation + '¡¿«»“”‘’'
```

Ahora tratamos como abrimos el archivo, en este caso con with open, luego indicamos el nombre del archivo, r para leer y además indicamos que use una codificación utf-8 para tratar correctamente el texto.

Cabe decir que con este método el archivo de texto tiene que estar en el mismo directorio que el código ya que solo así lo abrirá a no se que se le especifique la ruta o que se trate con una ventana de selección de archivo

```
with open('tu_me_quieres_blanca.txt', 'r', encoding='utf-8') as
archivo:
    contenido = archivo.read()
```

Como hemos usado en el ejercicio del nivel uno, creamos una “tabla” para usar de traducción donde le aplicamos nuestra nuevo metodo de punctuation

Y luego lo volvemos a convertir en texto.

```
tabla = str.maketrans('', '', puntuacion_extra)
texto_limpio =contenido.translate(tabla)
```

Lo separamos por palabras.

```
palabras = texto_limpio.split()
```

Definimos un diccionario vacío para almacenar los valores. Seguidamente iniciamos el bucle para iterar sobre cada palabra del texto.

Primeramente con capitalize, pondremos mayúscula la inicial de cada palabra.

Luego, condicionado con if le decimos si la palabra existe en el diccionario que hemos creado, sume una, sino que sea 1 como la primera.

```
contador = {}
for palabra in palabras:
    palabra = palabra.capitalize()

    if palabra in contador:
        contador[palabra] += 1
    else:
        contador[palabra] = 1
```

Para ordenar por orden alfabético. Iteramos sobre cada palabra y frecuencia dentro de los elementos de nuestra variable contador, donde la lavariable inicial le asignamos la posición 0 (primera) de la palabra. Si la inicial no esta en nuestra variable para almacenarla, se añade la inicial.

Por último, añadimos una variable para el total de las palabras.

```
contador_alfabetico = {}
for palabra, frecuencia in contador.items():
    inicial = palabra[0]
    if inicial not in contador_alfabetico:
        contador_alfabetico[inicial] = {}
    contador_alfabetico[inicial][palabra] = frecuencia

total=len(palabras)
```

Luego al usuario le decimos cuantas palabras hay en total con el primer print.

Imprimimos con un bucle para cada letra y palabra. De los elementos (ítems) de la variable contador_alfabetico, lo ordenamos (sorted) por orden de abecedario luego con un bucle anidado hacemos lo mismo para las palabras que van dentro de cada diccionario de cada inicial.

```
print(f"Hay un total de {total} palabras distribuidas de la siguiente
forma:")
for letra, palabras in sorted(contador_alfabetico.items()):
    print(f"{letra}:""{")
    for palabra, frecuencia in sorted(palabras.items()):
        print(f"  {palabra}: {frecuencia}")
    print("}")
```