



Gorka80 / S7_T7



<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main ▾ S7_T7 / Tasca_7_vf (amb comentaris i revisat).ipynb ▾

Go to file t ...

Gorka80 Add files via upload

6a0dc99 · 13 minutes ago

1089 lines (1089 loc) · 48.7 KB

Preview Code Blame

Raw

Exercici 1 (Nivell 1)

Calculadora de l'índex de masa corporal

Escriu una funció que calculi l'IMC ingressat per l'usuari/ària, és a dir, qui ho executi haurà d'ingressar aquestes dades. La funció ha de classificar el resultat en les respectives categories.

In [14]:

```
#La funció no té paràmetres d'entrada perquè es demanen per pantalla que l'usuari els entri.
def ingresar_peso_y_altura():

#El usuario entra el peso y la altura en kg y
#metros respectivamente y si no los entra bien se hacen ciertas acciones.

#1) Si el usuario se despista y entra el peso y la altura con decimales en vez de un punto,
#el programa corrige automáticamente el decimal a punto.

#2) Si el usuario entra textos, en el peso y la altura, da un error y devuelve None, None.

#3) Si el peso y la altura no son valores normales, el programa da error y se devuelve None, None.

try:
    peso_str=input("Entra tu peso en kg")
    peso_flo=float(peso_str)

except ValueError:
    #Corregión de decimal a punto
    peso_str_new=peso_str.replace(",",".")

    if peso_str_new==peso_str:
        print("¡Error! Se ha entrado", peso_str, "como peso que es un texto. Repetir el proceso.")
        return (None,None)

    peso_flo=float(peso_str_new)

try:
    altura_str=input("Entra tu altura en metros")
    altura_flo=float(altura_str)

except ValueError:
    #Corregión de decimal a punto
    altura_str_new=altura_str.replace(",",".")

    if altura_str_new==altura_str:
        print("¡Error! Se ha entrado", altura_str,"como altura que es un texto. Repetir el proceso.")
        return (None,None)

    altura_flo=float(altura_str_new)

if peso_flo<=0 or altura_flo>3 or altura_flo<=0:
    print ("¡Error! El peso", peso_flo, "kg", "introducido es negativo o la altura", altura_flo, "m", "es negativa o est"
          "Repetir el proceso.",sep=" ")
    return (None,None)

return (peso_flo,altura_flo)

#La función imc, se ha preparado para que solo se pueda ejecutar, tal como dice el enunciado,
#para que un usuario entre los datos del peso en kg y la altura en metros. Así todos los controles
#de error están encapsulados en la función ingresar_peso_y_altura

def cálculo_imc():

    (peso,altura)=ingresar_peso_y_altura()

    if (peso!=None and altura!=None):

        imc=round(peso/(altura**2),2)
        rango_imc=""

        if imc<18.5:
            clas_imc="Bajo peso (menos de 18,5)"

        elif imc>=18.5 and imc<=24.9:
            clas_imc="Peso normal (18,5-24,9)"

        elif imc>=25 and imc<=29.8:
            clas_imc="Sobrepeso (25-29,9)"

        else:
            clas_imc="Obesidad (más de 30)"

        print (f"El usuario de peso {peso} kg y altura {altura} m tiene un imc de {imc} kg/m2.\nSe consideraría como: \'{clas_imc}\'")
        return (imc,clas_imc)

    else:
        return (None,None)

###CÓDIGO PRINCIPAL###3

(imc,clas_imc)=cálculo_imc()
```

El usuario de peso 102.0 kg y altura 1.55 m tiene un imc de 42.46 kg/m².
Se consideraría como: "Obesidad (más de 30)"

Exercici 2 (Nivell 1)

Convertidor de temperaturas.

- Existeixen diverses unitats de temperatura utilitzades en diferents contextos i regions. Les més comunes són Celsius ($^{\circ}\text{C}$), Fahrenheit ($^{\circ}\text{F}$) i Kelvin (K). També existeixen altres unitats com Rankine ($^{\circ}\text{Ra}$) i Reaumur ($^{\circ}\text{Re}$). Selecciona almenys 2 conversors, de tal manera que en introduir una temperatura retorni, com a mínim, dues conversions.

In [2]:

```
#Quan volem imprimir els resultats, voldrem indicar els graus i les unitats.
#Com que l'usuari ens ha entrat digits de 1 a 5 per les unitats, necessitem tenir
#una llista de 5 elements i cada element una llista del nom i la unitat de la temperatura.
def list_T_nom_and_unid(T_nombres,T_unidades):
    #1. Celsius C, 2. Fahreneit F, 3. Kelvin K, 4. Rankine Ra, 5. Reaumier Re
    list_T_nom_unid=[]

    #Creo una llista de llistes, una llista de 5 elements de dos elements en cada element.
    for j in range(5):
        list_nom_unid=[None for i in range(2)]
        list_T_nom_unid.append(list_nom_unid)

    #Assigno els valors
    for i in range(5):
        for j in range(1):
            list_T_nom_unid[i][j]=T_nombres[i]
            list_T_nom_unid[i][j+1]=T_unidades[i]

    return (list_T_nom_unid)

#Si algún dels valors entrats no és correcta, retornem directament None,[] i així el programa no s'executarà.
def ingresar_temperatura_y_dos_conversores():

    list_convers=[]
    #Em farà servei la llista de todos_conversores, per comprovar si els valors entrats
    #de conversió són correctes.

    try:
        grados_entrada_str=input("Entra un valor con decimal con punto para una temperatura")
        grados_entrada=float(grados_entrada_str)

    except ValueError:
        print(f";Error! El valor entrado de {grados_entrada_str} no es correcto. Comprueba que sea un número decimal con punto")
        print("Repetir el proceso.")
        #Així m'asseguro de que si trobo un error, la funció ja ni demanarà els conversors de temperatura.
        return(None,[])

    try:
        dig_T_entrada_str=input("""Entra un número del 1 al 5 para indicar las unidades de la temperatura que has entrado.
                                         1. Celsius 2. Fahreneit 3. Kelvin 4. Rankine 5. Reaumur""")
        dig_T_entrada=int(dig_T_entrada_str)

        if dig_T_entrada not in range(1,6):
            print(f";Error! El valor entrado de {dig_T_entrada} no es correcto. Comprueba que has entrado un número del 1 al 5")
            print("Repetir el proceso.")
            return(None,[])

    except ValueError:
        print(f";Error! El valor seleccionado de unidad de entrada {dig_T_entrada_str} no es correcto. Comprueba que has entrado un número del 1 al 5")
        print("Repetir el proceso.")
        return (None,[])
    try:
        dig_T_salida_str_1=input("""Entra un número del 1 al 5 para indicar la primera unidad de conversión.
                                         1. Celsius 2. Fahreneit 3. Kelvin 4. Rankine 5. Reaumier Re""")
        list_convers.append(int(dig_T_salida_str_1))

        if list_convers[0] not in range(1,6):
            print(f";Error! El valor entrado del primer conversor {list_convers[0]} no es correcto. Comprueba que has entrado un número del 1 al 5")
            print("Repetir el proceso.")
            return(None,[])

    except ValueError:
        print(f";Error! El valor entrado para el primer conversor, {dig_T_salida_str_1} no es correcto. Debe ser un número del 1 al 5")
        print("Repetir el proceso.")
        return (None,[])
    try:
        dig_T_salida_str_2=input("""Entra un número del 1 al 5 para indicar la segunda unidad de conversión.
                                         1. Celsius 2. Fahreneit 3. Kelvin 4. Rankine 5. Reaumier Re""")
        list_convers.append(int(dig_T_salida_str_2))

        if list_convers[1] not in range(1,6):
            print(f";Error! El valor entrado del segundo conversor {list_convers[1]} no es correcto. Comprueba que has entrado un número del 1 al 5")
            print("Repetir el proceso.")
            return(None,[])

    except ValueError:
        print(f";Error! El valor entrado para el segundo conversor, {dig_T_salida_str_1} no es correcto. Debe ser un número del 1 al 5")
        print("Repetir el proceso.")
        return (None,[])
    return (grados_entrada,dig_T_entrada,list_convers)
```

```

#En funció de les unitats de la temperatura de entrada, tindrem 5 casuístiques per convertir-la.
#Per no complicar el codi cada casuística l'hem separat amb una funció.
def grados_entrada_a_otra(grados_entrada,dig_T_entrada,dig_T_destino):

    #1. Celsius C, 2. Fahrenheit F, 3. Kelvin K, 4. Rankine Ra, 5. Reaumur Re
    match dig_T_entrada:

        case 1:
            grados_destino=convers_celsius_a_otra(grados_entrada,dig_T_destino)
        case 2:
            grados_destino=convers_farenheit_a_otra(grados_entrada,dig_T_destino)
        case 3:
            grados_destino=convers_kelvin_a_otra(grados_entrada,dig_T_destino)
        case 4:
            grados_destino=convers_rankine_a_otra(grados_entrada,dig_T_destino)
        case 5:
            grados_destino=convers_reaumur_a_otra(grados_entrada,dig_T_destino)

    return grados_destino

def convers_celsius_a_otra(grados_entrada,dig_T_destino):

    #1. Celsius C, 2. Fahrenheit F, 3. Kelvin K, 4. Rankine Ra, 5. Reaumur Re
    match dig_T_destino:

        case 1:
            grados_destino=grados_entrada
        case 2:
            grados_destino=grados_entrada*9/5+32
        case 3:
            grados_destino=grados_entrada+273.15
        case 4:
            grados_destino=grados_entrada*9/5+491.67
        case 5:
            grados_destino=grados_entrada*0.8

    return (round(grados_destino,2))

def convers_farenheit_a_otra(grados_entrada, dig_T_destino):

    #1. Celsius C, 2. Fahrenheit F, 3. Kelvin K, 4. Rankine Ra, 5. Reaumur Re
    match dig_T_destino:
        case 1:
            grados_destino=(grados_entrada-32)*5/9
        case 2:
            grados_destino=grados_entrada
        case 3:
            grados_destino=(grados_entrada-32)*5/9+273.15
        case 4:
            grados_destino=grados_entrada+459.67
        case 5:
            grados_destino=(grados_entrada-32)/2.25000002

    return (round(grados_destino,2))

def convers_kelvin_a_otra(grados_entrada, dig_T_destino):

    #1. Celsius C, 2. Fahrenheit F, 3. Kelvin K, 4. Rankine Ra, 5. Reaumur Re
    match dig_T_destino:
        case 1:
            grados_destino=grados_entrada-273.15
        case 2:
            grados_destino=(grados_entrada-273.15)*9/5+32
        case 3:
            grados_destino=grados_entrada
        case 4:
            grados_destino=(grados_entrada-273.15)*1.8+491.67
        case 5:
            grados_destino=(grados_entrada-273.15)/1.25

    return (round(grados_destino,2))

def convers_rankine_a_otra(grados_entrada, dig_T_destino):

    #1. Celsius C, 2. Fahrenheit F, 3. Kelvin K, 4. Rankine Ra, 5. Reaumur Re
    match dig_T_destino:
        case 1:
            grados_destino=(grados_entrada-491.67)*5/9
        case 2:
            grados_destino=grados_entrada-459.67
        case 3:
            grados_destino=grados_entrada*5/9
        case 4:
            grados_destino=grados_entrada
        case 5:
            grados_destino=(grados_entrada-491.67)/2.25000002

    return (round(grados_destino,2))

def convers_reaumur_a_otra(grados_entrada,dig_T_destino):

    #1. Celsius C, 2. Fahrenheit F, 3. Kelvin K, 4. Rankine Ra, 5. Reaumur Re
    match dig_T_destino:
        case 1:
            grados_destino=grados_entrada*1.25
        case 2:
            grados_destino=(grados_entrada*2.25)+32
        case 3:
            grados_destino=(grados_entrada*1.25)+273.15
        case 4:
            grados_destino=grados_entrada-459.67
        case 5:
            grados_destino=(grados_entrada*5/9)-273.15

```

```

    case 4:
        grados_destino=(grados_entrada*2.25)+491.67
    case 5:
        grados_destino=grados_entrada

    return (round(grados_destino,2))

#####
#####Codi Principal#####
#####

T_nombres=["Celsius","Fahrenheit","Kelvin","Rankine","Reaumur"]
T_unidades=["C","F","K","Ra","Re"]

(grados_entrada,dig_T_entrada, list_convers)=ingresar_temperatura_y_dos_conversores()
list_T_nom_unid=list_T_nom_and_uni(T_nombres,T_unidades)
Sortida: [['Celsius', 'C'], ['Fahrenheit', 'F'], ['Kelvin', 'K'], ['Rankine', 'Ra'], ['Reaumur', 'Re']]

#Pot ser que l'usuari no recordi el que ha demanat. Amb el missatge ho pot comprovar tot, i fer la confirmació amb s o n.
mensaje=f'{La temperatura que ha entrat es de {grados_entrada} °{list_T_nom_unid[dig_T_entrada-1][1]} (grados {list_T_nom_unid[dig_T_entrada-1][0]})} (grados {list_T_nom_unid[dig_T_entrada-1][1]}) (grados {list_T_nom_unid[dig_T_entrada-1][0]}) equivalent a {list_T_nom_unid[dig_T_entrada-1][1]} {list_T_nom_unid[dig_T_entrada-1][0]}'
confirmacion=input(mensaje)
#Si algun dels valors d'entrada s'han entrat malament o no hem fet la confirmació anterior no entrarem aquí.
if grados_entrada!=None and len(list_convers)==2 and confirmacion=="s":

    grados_conv_1=grados_entrada_a_otra(grados_entrada,dig_T_entrada, list_convers[0])
    print(f'{grados_entrada} °{list_T_nom_unid[dig_T_entrada-1][1]} (grados {list_T_nom_unid[dig_T_entrada-1][0]}) equivalent a {list_T_nom_unid[dig_T_entrada-1][1]} {list_T_nom_unid[dig_T_entrada-1][0]}')

    grados_conv_2=grados_entrada_a_otra(grados_entrada,dig_T_entrada, list_convers[1])
    print(f'{grados_entrada} °{list_T_nom_unid[dig_T_entrada-1][1]} (grados {list_T_nom_unid[dig_T_entrada-1][0]}) equivalent a {list_T_nom_unid[dig_T_entrada-1][1]} {list_T_nom_unid[dig_T_entrada-1][0]}')

#No cal fer un else, doncs en el moment que hi hagi un error al fer l'ingrés de les dades, ja hi hagut una print de l'error
#on s'indicava que s'havia de repetir tot el procés i no entrariem en aquesta condició: grados_entrada!=None and len(list

```

20.0 °F (grados Fahrenheit) equivalen a 479.67 °Ra (grados Rankine)
20.0 °F (grados Fahrenheit) equivalen a -5.33 °Re (grados Reaumur)

Exercici 3 (Nivell 1)

Comptador de paraules d'un text.

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

```

In [5]: #Es tracta d'una funció que utilitzarem per anar substituint diferents
#tipus de caracters repetits per un espai, " ".
#Per exemple, si trobem al text, una separació de més de dos espais entre
#paraules, en les passarà a un espai.
#c, serà el caràcter que voldrem substituir per un espai. També ho utilitzarem
#pels salts de línia.
def subs_car_repetits_per_espai (texto, c):

    #c pot ser " ", o "\n", o bé "." o bé ";" o bé ","
    #Si c és un espai, " ", aleshores seria un espai de 15 posicions com a màxim.
    #Pel salt de línia \n com a molt 15 salts de línia.
    match c:

        case " ":
            rep=15
        case "\n":
            rep=15
        #Per ".", "," i ";" ho podríem fer servir.
        case _:
            rep=1

    for i in range(rep,0,-1):
        car_repetits=i*c
        #Pels espais " " superiors a un espai, amb la condició superior a 1 (texto.count(car_repetits)>0):
        # seria suficient. Es a dir, només ens interessaria substituir espais que tinguin més d'un'espais, ja
        #que si teneïm un espai entre paraules ja ens estaria bé.
        # Però en el cas dels salts de línia \n un únic salt de línia també
        # s'haurà de substituir per un espai. Així que fent >0 ens servei per tots els casos. Pel cas dels espais
        #d'un únic espai, simplement te'l substituirà pel mateix.
        if texto.count(car_repetits)>0:
            texto=texto.replace(car_repetits," ")
            #print(texto)

    return texto

#Donat un text en format molt general, amb forces casuístiques, et retorna una llista amb totes les
#paraules en cada posició de la llista
def paraules_text_a_llista(texto):

    signes_punt=[".",",",";","\n",")","!","!",";","?","?"]

    #print("TEXTO ORIGINAL\n\n")
    print(texto)

    #print("\n\nTEXTO CONVERTIDO EN MÍNUSCULAS\n\n")
    #Alhora de comptar paraules, volem que Està i està les compti com la mateixa,
    #ja que ho diferenciarà com paraules diferents sinó.
    texto_n=texto.lower()
    #print(texto_n)

    #print("\n\nTEXTO DESPRÉS DE CANVIAR SIGNES PUNTUACIÓ PER ESPAIS\n\n")

    for element in signes_punt:

```

```

--> ----->----->
    texto_n=texto_n.replace (element, " ")
    #print(texto_n)

    #print("\n\nTEXT DESPRÉS D'ELIMINAR TOTS ELS SALTS DE PÀGINA PER UN ESPAI\n\n")
    texto_n=subs_car_repetits_per_espai (texto_n, "\n")
    #print(texto_n)

    #Es millor fer-ho al final, ja que quan substituïm signes puntuació per espais, i s'eliminen els
    #salts de pàgina anterior per espais es poden formar espais superiors a un entre paraules. Per això
    #l'executa al final.
    #print("\n\nTEXT DESPRÉS D'ELIMINAR TOTS ELS ESPAIS SUPERIORS A UN ESPAI\n\n")
    texto_n=subs_car_repetits_per_espai (texto_n, " ")
    #print(texto_n)

    #Al afegir un texto_n=texto.replace (".", " "), em va molt bé per separar frases amb un punt i seguit
    #però al final del text apareix amb un espai que després me'l separarà a part com si fos una paraula.
    #Fent la tècnica del slice m'elimino aquest últim espai.
    texto_n=texto_n[:len(texto_n)-1]

    #Ara es veu el sentit de tot el que hem fet. Passem la cadena de text a una llista, separant els elements
    #que estan separats per un espai,#que són les paraules
    lista=texto_n.split(" ")
    #print(f"El text té {len(lista)} paraules")

    return lista

#Creem un diccionari d'una llista de paraules que poden estar repetides amb les paraules com a clau
# i amb el número de vegades que apareix cada paraula com a valor
def dict_paraules_and_count(list_paraules):

    dict_paraules_count=dict.fromkeys(list_paraules,0)

    #Manera vella d'inicialitzar el diccionari.
    # for paraula in list_paraules:
    #     dict_paraules_count[paraula]=0

    #Per cada paraula que estigui repetida de la llista,
    # s'incrementarà al valor de l'element del diccionari una unitat.
    for paraula in list_paraules:
        dict_paraules_count[paraula]+=1

    return dict_paraules_count

#####
#####Codi Principal#####
#####Per la última visualització de la solució#####
from tabulate import tabulate

texto="""Este es un texto de prueba. Algunas palabras tienen más De un espacio.

También existen frases separadas por varios saltos de página. Algunas frases
después del punto, no tienen espacios. Existen palabras separadas por coma sin
espacio, o también por punto y coma; claro.

Intentaremos resolver, este caso general."""

list_paraules=paraules_text_a_llista(texto)
list_count_x_ele=[]

#Per cada paraula de la list_paraules, coompto les vegades que apareix a la llista
#amb el mètode count i l'afegeixo a la llista list_count_x_ele.

dic_parau_count=dict_paraules_and_count(list_paraules)

# Faig pip install tabulate a msdos
#Visualitzaré el diccionari en dues columnes ja que així és més fàcil veure els resultats

tabla = [[clave, valor] for clave, valor in dic_parau_count.items()]
print(tabulate(tabla, headers=["Clave", "Valor"]))

```

Este es un texto de prueba. Algunas palabras tienen más De un espacio.

También existen frases separadas por varios saltos de página. Algunas frases
después del punto, no tienen espacios. Existen palabras separadas por coma sin
espacio, o también por punto y coma; claro.

Intentaremos resolver, este caso general.

Clave	Valor
este	2
es	1
un	2
texto	1
de	3
prueba	1
algunas	2
palabras	2
tienen	2
más	1
espacio	2
también	2
existen	2
frases	2
separadas	2
por	3
varios	1
saltos	1

página	1
después	1
del	1
punto	2
no	1
espacios	1
coma	2
sin	1
o	1
y	1
claro	1
intentaremos	1
resolver	1
caso	1
general	1

Exercici 4 (Nivell 1)

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 són únics. Si aquest no és el cas, la funció hauria d'imprimir un missatge d'avertiment.

In [8]:

```
#Donat un diccionari, ens es útil extreure directament una llista de les
#claus i una llista dels valors.
def llistes_claus_valors_diccionari(dicc):

    return(list(dicc.keys()),list(dicc.values()))

#Es una funció que retornarà true o false.
#Si la llista de valors, té algun element repetit, ja retornarà false.
def llista_valors_poden_ser_claus(llista_valors):

    pot_ser_clau=True

    #Ens va millor fer un for, ja que el mètode count de les llistes
    #va recorrent els elements de la llista, i per tant no podríem
    #utilitzar el count amb un index de la llista.
    for element in llista_valors:
        if llista_valors.count(element)>1:
            #Si trobem ja una condició en que hi ha valors repetits,
            #el return fa com de break, i ens estalviem de recorrer tota la llista
            #ja que sortim de la funció.
            return False
    #Si recorrem tots els elements de llista es que No hem entrat a la condició
    #que ens hauria retornat false, i per això retornaríem cert aquí.
    return pot_ser_clau

#Es directament la funció que ens demana l'enunciat
def diccionari_invers (enquesta):

    (list_claus,list_valors)=llistes_claus_valors_diccionari(enquesta)

    if llista_valors_poden_ser_claus(list_valors):
        print("\nEl diccionari presentat si pot passar els valors a claus d'un diccionari invers,\n ja que aquests no estan repetits.\n")
        #El zip ens fa únics la llista de valors per tal que després es pugui aplicar la funció dict.
        return(dict(zip(list_valors,list_claus)))
    else:
        print("\nEl diccionari presentat no pot passar els valors a claus d'un diccionari invers,\n ja que aquests estan repetits.\n")
        #No cal retornar None, doncs ell per defecte ho farà si no hi ha un return
        #return None

#####Codi principal#####
enuesta_1={("a":1,"b":2,"c":3,"d":4)}
enuesta_2={("a":1,"b":2,"c":2,"d":4)}

#Fem els dos exemples, un que sí que pot fer l'invers i l'altre que no.
print("diccionari_enuesta_1",enuesta_1,sep="")
enuesta_inversa_1=diccionari_invers(enuesta_1)
if enuesta_inversa_1!=None:
    print("\nEls resultats del diccionari invers són:")
    print("\ndiccionari_enuesta_inversa_1",enuesta_inversa_1,sep="")
print("\n#####\n")
print("diccionari_enuesta_2",enuesta_2,sep="")
enuesta_inversa_2=diccionari_invers(enuesta_2)
if enuesta_inversa_2!=None:
    print("\nEls resultats del diccionari invers són:")
    print("\ndiccionari_enuesta_inversa_2",enuesta_inversa_2,sep="")
```

diccionari_enuesta_1{'a': 1, 'b': 2, 'c': 3, 'd': 4}

El diccionari presentat sí pot passar els valors a claus d'un diccionari invers,
ja que aquests no estan repetits.

Els resultats del diccionari invers són:

diccionari_enuesta_inversa_1{1: 'a', 2: 'b', 3: 'c', 4: 'd'}

#####

```
diccionari_enquesta_2{ a : 1, b : 2, c : 2, d : 4}
```

El diccionari presentat no pot passar els valors a claus d'un diccionari invers, ja que aquests estan repetits.

Exercici 1 (Nivell 2)

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 aquest cas, en l'exercici anterior imprimies un missatge d'avertiment, ara, els valors del diccionari resultant hauran d'emmagatzemar-se com una llista. Tingues en compte que si és un valor únic no ha de ser una llista.

In [9]:

```
#Funció explicada anteriorment
def llistes_claus_valors_diccionari(dicc):

    return(list(dicc.keys()),list(dicc.values()))

#Funció explicada anteriorment
def dict_paraules_and_count(list_paraules):

    ##Manera eficient de construir un diccionari, amb una
    #llista de paraules que poden ser reptides, i de valor 0.
    dict_paraules_count=dict.fromkeys(list_paraules,0)

    #Manera ineficient d'inicialitzar el diccionari del codi anterior.
    # for paraula in list_paraules:
    #     dict_paraules_count[paraula]=0

    #Per cada paraula que estigui repetida de la llista,
    # s'incrementarà al valor de l'element del diccionari una unitat.
    for paraula in list_paraules:
        dict_paraules_count[paraula]+=1

    return dict_paraules_count

#Inicialitzo el diccionari invers, amb els valors com a clau,
#i els valors que estiguin reptits s'assignarà una llista buida.
def inicialitzar_revers_dict_plus(dict_paraules_cont):

    reverse_dict_plus={}

    for key in dict_paraules_cont.keys():
        if dict_paraules_cont[key]>1:
            reverse_dict_plus[key]=[]
        else:
            #Si els valors no són reptits, fent None, podem assignar
            #posteriorment un escalar o string.
            reverse_dict_plus[key]=None

    return reverse_dict_plus

def construir_reverse_dict_plus(dicc):

    reverse_dict_plus={}
    list_claus=[]
    list_valors=[]

    (list_claus,list_valors)=llistes_claus_valors_diccionari(dicc)

    dict_valors_i_frec=dict_paraules_and_count(list_valors)
    reverse_dict_plus=inicialitzar_revers_dict_plus(dict_valors_i_frec)

    for key in dicc.keys():

        #Només en el cas que el valor que hem trobat estigui repetit
        #assignarem la seva clau com una llista dintre la clau del nou
        #diccionari invers
        if dict_valors_i_frec[dicc[key]]>1:
            reverse_dict_plus[dicc[key]].append(key)

        else:
            reverse_dict_plus[dicc[key]]=key

    return reverse_dict_plus

### CODI PRINCIPAL #####
enquesta={({x:"apple",y:"banana",z:"banana"})}
enquesta_2={({x:"apple",y:"banana",z:"banana",q:"pera",p:"pera",r:"apple",m:"lemon"})}

reverse_dict_plus=construir_reverse_dict_plus(enquesta_2)
print(reverse_dict_plus)
```

```
{'apple': ['x', 'r'], 'banana': ['y', 'z'], 'pera': ['q', 'p'], 'lemon': 'm'}
```

Exercici 2 (Nivell 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',(2.1.4,'not-a-number'),[1,2,'3',3.4]]

```
In [10]: #Es una funció que donat un element, s'assigna a la llista de floats
#o de strings si passa la conversió. Utilitzem el try justament per
#si no passa la conversió de float, sabem per la excepció que s'haurà
#d'afegir a la llista de strings.
def asignar_elem_a_list_float_o_list_str(element,list_float,list_str):
    try:
        element_float=float(element)
        list_float.append(element_float)

    except ValueError:
        list_str.append(element)

#L'únic que hem de distingir si donat un element de la llista
#de l'enunciat pot ser una llista o una tupla.
#Si es tracta d'una tupla o una llista, hem de recorrer els seus
#elements
def tractar_element_llista(element_list,list_float,list_str):

    if type(element_list)==tuple or type(element_list)==list:
        for element in element_list:
            asignar_elem_a_list_float_o_list_str(element,list_float,list_str)
    else:
        #Aquí element_list serà un únic valor.
        asignar_elem_a_list_float_o_list_str(element_list,list_float,list_str)

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

```
list_float=[]
list_str=[]
#A l'enunciat els resultats es presenten com a llista de dos llistes.
list_clas=[[[],[]]]

for element_list in conversion:
    tractar_element_llista(element_list,list_float,list_str)

list_clas[0]=list_float
list_clas[1]=list_str

print("La llista conversion de l'enunciat l'hem classificat automàticament en dues llistes:")
print("Llista de classificació: ",list_clas)
```

La llista conversion de l'enunciat l'hem classificat automàticament en dues llistes:
Llista de classificació: [[1.3, 1000000000.0, 2.0, 1.0, 1.0, 2.0, 3.0, 3.4], ['one', 'seven', '3-1/2', 'not-a-number']]

Exercici 1 (Nivell 3)

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 a la Z. Per exemple, per a l'arxiu "tu_me_quieres_blanca.txt" la sortida esperada seria: (veure enunciat IT Academy)

```
In [11]: #Funció explicada anteriorment
def subs_car_repetits_per_espai(texto, c):

    #c pot ser " ", o "\n", o bé "." o bé ";" o bé ";"
    #Si c és un espai, " ", aleshores seria un espai de 15 posicions com a màxim.
    #Pel salt de línia \n com a molt 15 salts de línia.
    #Hem considerat 15 com un nombre arbitrari força elevat.
    match c:

        case " ":
            rep=15
        case "\n":
            rep=15
        #Per ".", ",", ";" ho podríem fer servir.
        case _:
            rep=1

    for i in range(rep,0,-1):
        car_repetits=i*c
        #Pels espais " " superiors a un espai, amb la condició superior a 1 (texto.count(car_repetits)>0):
        # seria suficient. Es a dir, només ens interessaria substituir espais que tinguin més d'un'espais, ja
        #que si teneim un espai entre paraules ja ens estaria bé.
        # Però en el cas dels salts de línia \n un únic salt de línia també
        # s'haurà de substituir per un espai. Així que fent >0 ens servei per tots els casos. Pel cas dels espais
        #d'un únic espai, simplement te'l substituirà pel mateix.
        if texto.count(car_repetits)>0:
            texto=texto.replace(car_repetits," ")
            #print(texto)

    return texto

#Funció explicada anteriorment
def paraules_text_a_llista(texto):

    #print("TEXTO ORIGINAL\n\n")
    #print(texto)

    #Mitjançant una llibreria d'expressions regulars, podríem haver-ne inclòs més,
    #però considerem que per aquest exercici era suficient.
    signes_punt=[".",";",";","(",")","!","|",":"]
```

```

#print("\n\nTEXTO CONVERTIDO EN MÍNUSCULAS\n\n")
#A l' hora de comptar paraules, volem que Està i està les contagi com la mateixa,
#ja que ho diferenciarà com paraules diferents sinó.
texto_n=texto.lower()
#print(texto_n)

#print("\n\nTEXTO DESPRÉS DE CANVIAR SIGNES PUNTUACIÓ PER ESPAIS\n\n")

for element in signes_punt:
    texto_n=texto_n.replace (element, " ")

#print(texto_n)

#print("\n\nTEXT DESPRÉS D'ELIMINAR TOTS ELS SALTS DE PÀGINA PER UN ESPAI\n\n")
texto_n=subs_car_repetits_per_espai (texto_n, "\n")
#print(texto_n)

#Es millor fer-ho al final, ja que quan substituïm signes puntuació per espais, i s'eliminen els
#salts de página anterior per espais es poden formar espais superiors a un entre paraules. Per això
#l'executo al final.
#print("\n\nTEXT DESPRÉS D'ELIMINAR TOTS ELS ESPAIS SUPERIORES A UN ESPAI\n\n")
texto_n=subs_car_repetits_per_espai (texto_n, " ")
#print(texto_n)

#AL afegir un texto_n=texto.replace (".", " "), em va molt bé per separar frases amb un punt i seguit
#però al final del text apareix amb un espai que després me'l separarà a part com si fos una paraula.
#Fent la tècnica del slice m'elimino aquest últim espai.
texto_n=texto_n[:len(texto_n)-1]

#Ara es veu el sentit de tot el que hem fet. Passem la cadena de text a una llista, separant els elements
#que estan separats per un espai,
#que són les paraules
list_paraules=texto_n.split(" ")
#print(f"El text té {len(list Paraules)} paraules")

return list_paraules

#Funció explicada anteriorment
def llistes_claus_valors_diccionari(dicc):

    return(list(dicc.keys()),list(dicc.values()))

#Versió de codi antiga
# for key in dicc.keys():
#     list_claus.append(key)
#     list_valors.append(dicc[key])

# return (list_claus,list_valors)

#Funció explicada anteriorment
def dict_paraules_and_count(list_paraules):

    ##Manera eficient de construir un diccionari, amb una
    #llista de paraules que poden ser reptides, i de valor 0.
    dict_paraules_count=dict.fromkeys(list_paraules,0)

    #Manera ineficient d'inicialitzar el diccionari del codi anterior.
    # for paraula in list_paraules:
    #     dict_paraules_count[paraula]=0

    #Per cada paraula que estigui repetida de la llista,
    # s'incrementarà al valor de l'element del diccionari una unitat.
    for paraula in list_paraules:
        dict_paraules_count[paraula]+=1

    return dict_paraules_count

#Si l' últim element, té la mateixa primera lletra que l'anterior,
# hem d'afegir aquest últim element al dic_aux que teniem, ja que en l'anterior bucle
# havíem afegit a dic_aux l'element de la posició len(list_claus)-2) i no pas de la última,
#la len(list_claus)-1.
def tractar_ultima_clau(list_claus,list_valors,dic_aux,dic_sol):

    i_ult_ele=len(list_claus)-1

    if list_claus[i_ult_ele][0]==list_claus[i_ult_ele-1][0]:
        dic_aux[list_claus[i_ult_ele]]=list_valors[i_ult_ele]
        dic_sol[list_claus[i_ult_ele]][0]=dic_aux
    #Si justament l'últim element de la llista canvia de lletra, aleshores hem
    # de ressestinar dic_aux, i afegir l'últim element a dic_aux
    else:
        #He de posar la clau de la solució i-1, ja que l'última posició, la i
        #té una lletra diferent.
        #dic_sol[list_claus[i-1][0]]=dic_aux
        dic_aux={}
        #Ara he d'afegir a dic_aux buit l'última clau i valor.
        dic_aux[list_claus[i_ult_ele]]=list_valors[i_ult_ele]
        #Assigno a l'última lletra del diccionari, la solució dic_aux
        dic_sol[list_claus[i_ult_ele]][0]=dic_aux

#####
#####Codi principal #####
#####

ruta_archivo="tu_me_quieres_blanca.txt"

```

```

wtn open(ruta_archivo, 'r', encoding='utf8') as archivo:
    texto=archivo.read()
#print(texto)

list_paraules=paraules_text_a_llista(texto)

#Creo el diccionari de paraules i la seva repetició.

#Tinc el diccionari de paraules repetides del text
dic_parau_count=dict_paraules_and_count(list_paraules)

#ordenem el diccionari per claus ascendents, pero la sortida de sorted es una llista de tuplas. Aleshores
#hem de tornar a fer dict per tornar a tenir un diccionari.
dic_parau_count_sorted = dict(sorted(dic_parau_count.items()))

#Per com ho farem ens va molt millor treballar en dues llistes,
#una que té les claus i l'altre que té la freqüència.
(list_claus,list_valors)=llistes_claus_valors_diccionari(dic_parau_count_sorted)

#Podriem posar el següent codi en una funció, però considerem que és força
#específic per aquesta solució de l'exercici, que és obtenir un diccionari
#on les claus son les lletres de l'abecedari i els valors són altres diccionaris,
#que contenen les paraules del text de la lletra de la clau i la seva freqüència.

#print(len(list_claus))

i=0
dic_sol={}

#Es un diccionari auxiliar que ens servirà per afegir els subdiccionaris
dic_aux={}

#Ens va millor treballar amb index
#Recorrem tota la llista de claus, fins la len(list_claus)-2.
#Recordem si la longitud de la llista es len(list_claus), l'últim element estarà a la posició len(list_claus)-1,
#ja que comencem per 0. Quan i=len(list_claus)-2, la condició list_claus[i][0]==list_claus[i+1][0], farà que la part
#dreta del membre arribem a la última posició de la llista i+1=len(list_claus)-1, i per tant no tindrem error.
#Per tant si i<=(len(list_claus)-1), al arribar a l'últim element després sortírem de l'índex de la llista.

while i<=(len(list_claus)-2):

    if list_claus[i][0]==list_claus[i+1][0]:
        dic_aux[list_claus[i]]=list_valors[i]
    else:
        #En el canvi de lletra de dos elements consecutius de la llista
        #no entro a la condició anterior, però he de guardar la paraula a dic_aux
        dic_aux[list_claus[i]]=list_valors[i]
        #Al diccionari solució, afageixo com a clau, la primera lletra, i vom a valor el diccionari auxiliar.
        dic_sol[list_claus[i][0]]=dic_aux
        #Si he entrat a l'else és que la primera lletra del següent element de la llista
        #de claus és diferent. Per tant, he de ressaltar el dic_aux.
        dic_aux={}
    i+=1

#Quan estiguem aquí és justament quan i=(len(list_claus)-1), que és l'últim element que ens queda per tractar
#del bucle anterior.
else:
    tractar_ultima_clau(list_claus,list_valors,dic_aux,dic_sol)

print(dic_sol)

```