sprint7

May 5, 2025

1 SPRINT 7

1.1 Nivel 1

1.1.1 Ejercicio 1

Calculadora Índice Masa Corporal Se solicita al usuario mediante input que introduzcan un valor de peso y altura.

Mediante bucles *while* y *try-except*, se repite este proceso hasta que el valor es válido (es decir, puede convertirse al tipo de dato *float*).

Se calcula el *imc* a través de su fórmula matemática y se calcula una de las categorías en base a este valor (con condicionales *if-else*).

Con el parámetro opcional *print_results* puede solicitarse que se impriman los resultados por pantalla.

Finalmente, la función devuelve un diccionario con el valor IMC y la categoría calculada.

[1]:
```
def calcular_imc(print_results = False):
    '''
    Calcula el Indice de Masa Corporal (IMC) para una altura y peso␣
 ↪introducidos por el usuario.
    Devuelve un diccionario con IMC y una de las cuatro categorías de IMC.
    '''

    retry_peso = True
    while retry_peso:
        try:
            peso = abs(float(input("Introduce tu peso en kilogramos: ")))
            retry_peso = False
        except:
            print("\n(!) El valor introducido no es válido.\n")

    retry_altura = True
    while retry_altura:
        try:
            altura_en_centimetros = abs(float(input("Introduce tu altura en␣
 ↪centímetros: ")))
            retry_altura = False
```

```
        except:
            print("\n(!) El valor introducido no es válido.\n")

    altura_en_metros = altura_en_centimetros / 100
    imc = round(peso/(altura_en_metros**2),2)

    if imc < 18.5:
        categoria = "bajo peso"
    elif imc < 25:
        categoria = "peso normal"
    elif imc < 30:
        categoria = "sobrepeso"
    else:
        categoria = "obesidad"

    if print_results:
        print(f"""
                ------------------------------
                Peso:      {peso} kg
                Altura:    {int(altura_en_centimetros)} cm
                IMC:       {imc}
                Categoría: {categoria.capitalize()}
                ------------------------------
                """)

    return {"IMC":imc, "Categoría": categoria}
```

Se ejecuta la función sin ninún parámetro.

[2]:
```
calcular_imc()
```

```
Introduce tu peso en kilogramos:  70
Introduce tu altura en centímetros:  180
```

[2]:
```
{'IMC': 21.6, 'Categoría': 'peso normal'}
```

Se puede solicitar que se muestren los resultados por pantalla con el parámetro **print_results**.

[3]:
```
calcular_imc(print_results = True)
```

```
Introduce tu peso en kilogramos:  95
Introduce tu altura en centímetros:  145


                ------------------------------
                Peso:      95.0 kg
                Altura:    145 cm
                IMC:       45.18
                Categoría: Obesidad
```

---

[3]: {'IMC': 45.18, 'Categoría': 'obesidad'}

1.2 Nivel 1

1.2.1 Ejercicio 2

Conversor de temperaturas Se definen tres funciones para convertir temperaturas en función de la unidad de temperatura de la que se parte: - convertir_celsius - convertir_kelvin - convertir_fahrenheit

La función **conversor_temperaturas** muestra un menu que permite escoger la unidad.

A través de un bucle *while*, esta selección se repite hasta que el valor introducido por el usuario es uno de los valores posibles mostrados en el menú. Los valores pueden introducire en mayúsculas o minúsculas, ya que son convertidos a minúsculas con el método *lower*.

En el caso de querer finalizar el programa sin ingresar ninguna temperatura, se puede escoger la opción "X". El programa devuelve un booleano *False* y termina.

Si se introduce una unidad, se solicita un valor de temperatura. Después, en base a la unidad previamente escogida, la función llama a una de las tres funciones conversoras que se habían declarado previamente y devuelve un diccionario conteniendo la temperatura en tres unidades.

Opcionalmente, a través del parámetro *print_result*, pueden mostrarse los valores por pantalla de un modo más legible. Se utiliza un bucle *for* para recorrer el diccionario y mostrar cada una de las entradas y la función *round* para reducir el número de decimales mostrados.

[4]:
```
def convertir_celsius(temperatura_celsius):
    '''Dada una temperatura en grados celsius, se devuelve un diccionario con␣
 ↪esta temperatura en ºC, ºF y ºK'''
    temperatura_fahrenheit = temperatura_celsius * (9/5) + 32
    temperatura_kelvin = temperatura_celsius + 273.15
    return {"C": float(round(temperatura_celsius,2)),
        "F": float(round(temperatura_fahrenheit,2)),
        "K": float(round(temperatura_kelvin,2))}


def convertir_kelvin(temperatura_kelvin):
    '''Dada una temperatura en grados kelvin, se devuelve un diccionario con␣
 ↪esta temperatura en ºC, ºF y ºK'''
    temperatura_celsius = temperatura_kelvin - 273.15
    temperatura_fahrenheit = (9/5) * (temperatura_kelvin - 273.15) + 32
    return {"C": float(round(temperatura_celsius,2)),
            "F": float(round(temperatura_fahrenheit,2)),
            "K": float(round(temperatura_kelvin,2))}
```

```
def convertir_fahrenheit(temperatura_fahrenheit):
    '''Dada una temperatura en grados fahrenheit, se devuelve un diccionario␣
 ↪con esta temperatura en ºC, ºF y ºK'''
    temperatura_celsius = (temperatura_fahrenheit - 32) * (5/9)
    temperatura_kelvin = (5/9)*(temperatura_fahrenheit - 32) + 273.15
    return {"C": float(round(temperatura_celsius,2)),
        "F": float(round(temperatura_fahrenheit,2)),
        "K": float(round(temperatura_kelvin,2))}


def conversor_temperaturas(print_result = False):
    '''
    Permite escoger una unidad de temperatura y un valor de temperatura en esa␣
 ↪unidad.
    Devuelve un diccionario con la temperatura expresada en grados Celsius,␣
 ↪Fahrenheit y Kelvin.
    '''

    reintenta_eleccion = True
    while reintenta_eleccion:
        print("¿En qué unidad está la temperatura a convertir?\n")
        print("  C - Para grados Celsius.")
        print("  F - Para grados Fahrenheit.")
        print("  K - Para grados Kelvin.")
        print("  X - Para Salir.\n")
        unidad_introducida = input("Introduce tu elección: ")

        if unidad_introducida.lower() in ["c", "f", "k", "x"]:
            reintenta_eleccion = False
        else:
            print("\n(!) La opción seleccionada no es válida.")
            print("Introduce uno de los valores presentados. (C, F, K, X)\n")

    if unidad_introducida == "x":
        print("\nEl programa ha finalizado.")
        return False
    else:
        reintenta_temperatura = True
        while reintenta_temperatura:
            try:
                temperatura = float(input("\nIntroduce el valor de temperatura␣
 ↪a convertir: "))
                print()
                reintenta_temperatura = False
            except:
                print("\n(!) El valor de temperatura no es válido.\n")
```

```
    if unidad_introducida == "c":
        temperaturas_convertidas = convertir_celsius(temperatura)
    elif unidad_introducida == "f":
        temperaturas_convertidas = convertir_fahrenheit(temperatura)
    elif unidad_introducida == "k":
        temperaturas_convertidas = convertir_kelvin(temperatura)
    else:
        return False

    if print_result:
        print("----------------------")
        for clave in temperaturas_convertidas:
            print(f"     {round(temperaturas_convertidas[clave],1)}  º{clave}")
        print("----------------------")

    return temperaturas_convertidas
```

[5]:
```
conversor_temperaturas()
```

```
¿En qué unidad está la temperatura a convertir?

  C - Para grados Celsius.
  F - Para grados Fahrenheit.
  K - Para grados Kelvin.
  X - Para Salir.


Introduce tu elección:  c

Introduce el valor de temperatura a convertir:  25
```

[5]:
```
{'C': 25.0, 'F': 77.0, 'K': 298.15}
```

Opcionalmente, mostrando los resultados por pantalla de manera más legible:

[6]:
```
conversor_temperaturas(print_result = True)
```

```
¿En qué unidad está la temperatura a convertir?

  C - Para grados Celsius.
  F - Para grados Fahrenheit.
  K - Para grados Kelvin.
  X - Para Salir.


Introduce tu elección:  k
```

```
Introduce el valor de temperatura a convertir:  305


---------------------
     31.9  ºC
     89.3  ºF
     305.0  ºK
---------------------
```

[6]:
```
{'C': 31.85, 'F': 89.33, 'K': 305.0}
```

1.3 Nivel 1

1.3.1 Ejercicio 3

Para contar las palabras en un texto, se han definido dos funciones distintas:

- **limpiar_texto**: convierte el texto en un string únicamente con caracteres alfabéticos y espacios.
- **contador_palabras**: cuenta las palabras de un texto. Utiliza la función anterior para preparar el texto.

limpiar_texto comprueba (en el marco de la función *try*) si es posible convertir el input dado al tipo de dato *string*. Si es posible, recorre cada uno de los caracteres de la cadena optando por añadirlos a una nueva cadena (llamada *texto_limpio*) en caso de ser caracteres **alfabéticos**. En el caso de valores **no alfabéticos**, estos son sustituidos por espacios. Nótese que en el caso de haber más de un carácter no alfabético, se añade únicamente **un solo espacio** a la cadena *texto_limpio*. Una vez se han comprobado todos los caracteres, la función devuelve la nueva cadena *texto_limpio*.

contador_de_palabras acepta un texto como parámetro. Se apoya en la ya presentada función *limpiar_texto*. Utilizando el método **split** se trocea el texto creando una **lista** de palabras. Crea un diccionario vacío *frecuencia_palabras* y recorriendo la lista creada con un bucle *for*, añade cada palabra como clave si no existe previamente, o suma uno al conteo de esa palabra en caso de que haya aparecido con anterioridad. Por defecto, el diccionario de frecuencias resultante se ordena alfabéticamente a través de una *dict comprehension*, este comportamiento puede controlarse con el parámetro opcional *orden_alfabetico*).

[7]:
```
def limpiar_texto(texto):
    '''
    El parámetro texto debe ser una cadena de caracteres (string).
    Devuelve el texto conservando únicamen caracteres alfabéticos y espacios.
    '''
    texto_limpio = ""
    try:
        texto = str(texto)
        for caracter in texto:
            if not caracter.isalpha():
                caracter = " "
```

```
                if (len(texto_limpio) > 0) and (texto_limpio[-1] != " "):
                    texto_limpio += caracter
            else:
                texto_limpio += caracter.lower()
    except:
        return False

    return texto_limpio


def contador_palabras(texto, orden_alfabetico = True):
    texto = limpiar_texto(texto)
    lista_palabras = texto.split()
    frecuencia_palabras = {}
    for palabra in lista_palabras:
        if palabra in frecuencia_palabras:
            frecuencia_palabras[palabra] += 1
        else:
            frecuencia_palabras[palabra] = 1

    if orden_alfabetico: # Por defecto, se devuelve el diccionario con las␣
 ↪claves ordenadas alfabeticamente
        frecuencia_palabras = {clave : frecuencia_palabras[clave] for clave in␣
 ↪sorted(frecuencia_palabras)}

    return frecuencia_palabras
```

Se declara un texto de prueba:

[8]:
```
texto_de_prueba = """
Tú me quieres alba,
me quieres de espumas,
me quieres de nácar.
Que sea azucena
Sobre todas, casta.
De perfume tenue.
Corola cerrada .

Ni un rayo de luna
filtrado me haya.
Ni una margarita
se diga mi hermana.
Tú me quieres nívea,
tú me quieres blanca,
tú me quieres alba.

Tú que hubiste todas
```

las copas a mano,
de frutos y mieles
los labios morados.
Tú que en el banquete
cubierto de pámpanos
dejaste las carnes
festejando a Baco.
Tú que en los jardines
negros del Engaño
vestido de rojo
corriste al Estrago.

Tú que el esqueleto
conservas intacto
no sé todavía
por cuáles milagros,
me pretendes blanca
(Dios te lo perdone),
me pretendes casta
(Dios te lo perdone),
¡me pretendes alba!

Huye hacia los bosques,
vete a la montaña;
límpiate la boca;
vive en las cabañas;
toca con las manos
la tierra mojada;
alimenta el cuerpo
con raíz amarga;
bebe de las rocas;
duerme sobre escarcha;
renueva tejidos
con salitre y agua:

Habla con los pájaros
y lévate al alba.
Y cuando las carnes
te sean tornadas,
y cuando hayas puesto
en ellas el alma
que por las alcobas
se quedó enredada,
entonces, buen hombre,
preténdeme blanca,
preténdeme nívea,
preténdeme casta.

```
"""
```

Se ejecuta la función pasando la cadena *texto_de_prueba* como argumento:

[9]:
```
contador_palabras(texto_de_prueba)
```

[9]:
```
{'a': 3,
 'agua': 1,
 'al': 2,
 'alba': 4,
 'alcobas': 1,
 'alimenta': 1,
 'alma': 1,
 'amarga': 1,
 'azucena': 1,
 'baco': 1,
 'banquete': 1,
 'bebe': 1,
 'blanca': 3,
 'boca': 1,
 'bosques': 1,
 'buen': 1,
 'cabañas': 1,
 'carnes': 2,
 'casta': 3,
 'cerrada': 1,
 'con': 4,
 'conservas': 1,
 'copas': 1,
 'corola': 1,
 'corriste': 1,
 'cuando': 2,
 'cubierto': 1,
 'cuerpo': 1,
 'cuáles': 1,
 'de': 8,
 'dejaste': 1,
 'del': 1,
 'diga': 1,
 'dios': 2,
 'duerme': 1,
 'el': 4,
 'ellas': 1,
 'en': 4,
 'engaño': 1,
 'enredada': 1,
 'entonces': 1,
```

```
'escarcha': 1,
'espumas': 1,
'esqueleto': 1,
'estrago': 1,
'festejando': 1,
'filtrado': 1,
'frutos': 1,
'habla': 1,
'hacia': 1,
'haya': 1,
'hayas': 1,
'hermana': 1,
'hombre': 1,
'hubiste': 1,
'huye': 1,
'intacto': 1,
'jardines': 1,
'la': 3,
'labios': 1,
'las': 7,
'lo': 2,
'los': 4,
'luna': 1,
'lévate': 1,
'límpiate': 1,
'mano': 1,
'manos': 1,
'margarita': 1,
'me': 10,
'mi': 1,
'mieles': 1,
'milagros': 1,
'mojada': 1,
'montaña': 1,
'morados': 1,
'negros': 1,
'ni': 2,
'no': 1,
'nácar': 1,
'nívea': 2,
'perdone': 2,
'perfume': 1,
'por': 2,
'pretendes': 3,
'preténdeme': 3,
'puesto': 1,
'pájaros': 1,
```

```
'pámpanos': 1,
'que': 6,
'quedó': 1,
'quieres': 6,
'rayo': 1,
'raíz': 1,
'renueva': 1,
'rocas': 1,
'rojo': 1,
'salitre': 1,
'se': 2,
'sea': 1,
'sean': 1,
'sobre': 2,
'sé': 1,
'te': 3,
'tejidos': 1,
'tenue': 1,
'tierra': 1,
'toca': 1,
'todas': 2,
'todavía': 1,
'tornadas': 1,
'tú': 8,
'un': 1,
'una': 1,
'vestido': 1,
'vete': 1,
'vive': 1,
'y': 5}
```

1.4 Nivel 1

1.4.1 Ejercicio 4

La función **corrige_diccionarios** intercambia las claves y los valores de un diccionario dado.

Para este ejercicio, no se aceptan diccionarios con valores duplicados (que generarían claves duplicadas), por lo que se le atribuye un valor **False al parámetro *acepta_duplicados***.

Para hacer este intercambio de claves y valores, se extraen los valores mediante el método de diccionario **values()** y se guardan en la variable claves como una lista. Las claves se extraen a través del método **keys()** y se guardan como lista en valores.

Se compara la longitud de la lista claves, con la misma lista convertida a **set** para comprobar si hay duplicados (el tipo de datos *set* no permite valores repetidos, por lo que la longitud debería diferir si existieran duplicados).

Si no existen repeticiones en las claves, se crea un nuevo diccionario con las listas **claves** y **valores**, en conjunción con la función *zip* utilizando un *dict_comprehension*.

Si existen repeticiones, dado que se solicita en el enunciado que no se acepten duplicados, se muestra un mensaje de error y se sale de la función devolviendo un bool False.

[10]:
```
def corrige_diccionario(diccionario_inicial, acepta_duplicados = True):
    '''
    Dado un diccionario inicial, se devuelve un diccionario con las claves y␣
 ↪los valores intercambiados.
    Se puede configurar si se aceptan diccionarios con valores duplicados o no-␣
 ↪En caso afirmativo, se devuelven listas
    asociadas a las claves duplicadas. En caso negativo, se devuelve un mensaje␣
 ↪de error si existen duplicados.
    '''
    claves = list(diccionario_inicial.values())
    valores = list(diccionario_inicial.keys())

    if len(claves) == len(set(claves)):
        diccionario = {clave:valor for (clave, valor) in zip(claves, valores)}

    else:
        if not acepta_duplicados:
            print("Error: multiple keys for one value")
            return False
        diccionario = {}
        for i in range(len(claves)):
            clave = claves[i]
            valor = valores[i]

            if claves.count(clave) == 1:
                diccionario[clave] = valor
            else:
                if clave not in diccionario:
                    diccionario[clave] = [valor]
                else:
                    diccionario[clave].append(valor)

    return diccionario
```

Se ejecuta la función pasando un diccionario **sin** valores duplicados:

[11]:
```
ejemplo1 = {1: "uno", 2:"dos", 3:"tres", 4:"cuatro", 5:"cinco", 6:"seis"}

corrige_diccionario(ejemplo1, acepta_duplicados = False)
```

[11]:
```
{'uno': 1, 'dos': 2, 'tres': 3, 'cuatro': 4, 'cinco': 5, 'seis': 6}
```

Se vuelve a ejecutar la función, en esta ocasión *con* valores duplicados:

[12]:
```
ejemplo2 = {1: "uno", 2:"dos", 3:"tres", 4:"cuatro", 5:"uno", 6:"dos"}

corrige_diccionario(ejemplo2, acepta_duplicados = False)
```

```
Error: multiple keys for one value
```

[12]:
```
False
```

1.5 Nivel 2

1.5.1 Ejercicio 1

Se utiliza la misma función que para el ejercicio anterior, pero en este caso se pasa el parámetro opcional para **aceptar duplicaods** (en caso de no pasar ningún parámetro, la opción por defecto es aceptar duplicados).

La lógica aplicada es la misma que en el ejercicio anterior excepto cuando se encuentran claves duplicadas: - Si la clave **no tiene duplicados**, el valor se guarda con su tipo de dato. - Si la clave **tiene duplicados, la primera vez** que se encuentra esa clavea al recorrer la lista de claves, se crea una lista que contenta el valor asociado a dicha clave. - Si la clave **tiene duplicados, y ya existe** esa clave en el diccionario, se añade el valor asociado a esa clave, a la lista ya existente (mediante el método *append*).

[13]:
```
ejemplo3 = {1: "uno", 2:"dos", 3:"tres", 4:"cuatro", 5:"uno", 6:"dos"}

corrige_diccionario(ejemplo3, acepta_duplicados = True)
```

[13]:
```
{'uno': [1, 5], 'dos': [2, 6], 'tres': 3, 'cuatro': 4}
```

1.6 Nivel 2

1.6.1 Ejercicio 2

La función ***conversion*** recorre una lista de uno o dos niveles de profundidad e intenta convertir cada uno de los valores al tipo de datos float.

Para recorrer la lista dada, se utiliza un bucle *for*. Si alguno de los elementos de la lista es una lista o tuple, se recorre este segundo nivel usando un segundo bucle. Con el fin de comprobar que se trata de una lista o tupla, se utiliza la función *isinstance*.

Para cada elemento se intenta (dentro de un bloque *try*) convertir el mismo al tipo de datos *float*. Si se consigue convertir, se añade el elemento a la lista *si_float* y si no se consigue, a la lista *no_float*.

Finalmente, se devuelve una tupla con las dos listas.

[14]:
```
def conversion(lista):
    '''
```

```
    Convierte una lista con uno o dos niveles de profundidad en una tupla con␣
↪dos listas, la primera con los valores que se
    han podido convertir al tipo de datos float y la segunda con aquellos␣
↪valores que no se han podido convertir.
    '''
    si_float = []
    no_float = []
    for item in lista:
        try:
            si_float.append(float(item))
        except:
            if isinstance(item, (list, tuple)):
                for item_en_sublista in item:
                    try:
                        si_float.append(float(item_en_sublista))
                    except:
                        no_float.append(item_en_sublista)
            else:
                no_float.append(item)
    return (si_float, no_float)
```

[15]:
```
mi_lista = [ '1.3', 'one' , '1e10' , 'seven', '3-1/2', ('2',1,1.
↪4,'not-a-number'), [1,2,'3','3.4']]

conversion(mi_lista)
```

[15]:
```
([1.3, 10000000000.0, 2.0, 1.0, 1.4, 1.0, 2.0, 3.0, 3.4],
 ['one', 'seven', '3-1/2', 'not-a-number'])
```

1.7 Nivel 3

1.7.1 Ejercicio 1

Se definen dos funciones nuevas:

- **lector_texto**: lee el contenido de un archivo *txt* con el nombre facilitado como parámetro de la función.
  - Se utiliza *open with* para generar un nuevo contexto (apertura y cierre del archivo tras la lectura).
  - El archivo se abre en *modo lectura ("r")*, y se lee el contenido del mismo con el método *read()*.
  - Se devuelve un único *string* con todo el contenido del archivo txt.
- **alfabetizar_diccionarios**: dado un diccionario con palabras como clave y su frecuencia de aparición el texto como valores, devuelve un diccionario con cada una de las letras iniciales como clave, conteniendo diccionarios con palabra como clave y su frecuencia como valor.

Se reutilizan las funciones definidas en el **Nivel 1 Ejercicio 3**: - **limpiar_texto**: descrita ante-

riormente. - **contador palabras**: descrita anteriormente.

[16]:
```
def limpiar_texto(texto):
    '''
    El parámetro texto debe ser una cadena de caracteres (string).
    Devuelve el texto conservando únicamen caracteres alfabéticos y espacios.
    '''
    texto_limpio = ""
    try:
        texto = str(texto)
        for caracter in texto:
            if not caracter.isalpha():
                caracter = " "
                if (len(texto_limpio) > 0) and (texto_limpio[-1] != " "):
                    texto_limpio += caracter
            else:
                texto_limpio += caracter.lower()
    except:
        return False

    return texto_limpio


def contador_palabras(texto, orden_alfabetico = True):
    '''
    Dado un texto previamente tratado con la función limpiar_texto,
    devuelve un diccionario con palabras como claves y su frecuencia de␣
 ↪aparición como valores.
    '''
    texto = limpiar_texto(texto)
    lista_palabras = texto.split()
    frecuencia_palabras = {}
    for palabra in lista_palabras:
        if palabra in frecuencia_palabras:
            frecuencia_palabras[palabra] += 1
        else:
            frecuencia_palabras[palabra] = 1

    if orden_alfabetico: # Por defecto, se devuelve el diccionario con las␣
 ↪claves ordenadas alfabeticamente
        frecuencia_palabras = {clave : frecuencia_palabras[clave] for clave in␣
 ↪sorted(frecuencia_palabras)}

    return frecuencia_palabras


def lector_txt(archivo_txt):
```

```
    '''
    Se pasa como parámetro un nombre de archivo txt.
    Devuelve un string con el contenido del archivo facilitado.
    '''
    with open(archivo_txt, "r") as contenido_txt:
        texto = contenido_txt.read()
    return texto


def alfabetizar_diccionarios(diccionario):
    '''
    Dado un diccionario de frecuencias (generado con la función␣
 ↪contador_palabras),
    devuelve un diccionario con las iniciales de cada palabra y diccionarios de␣
 ↪frecuencias para las palabras
    que empiezan cada inicial como valores.
    '''
    diccionario_por_inicial =  {}
    for clave in diccionario:
        inicial = clave[0]
        if inicial not in diccionario_por_inicial:
            diccionario_por_inicial[inicial] = {clave: diccionario[clave]}
        else:
            diccionario_por_inicial[inicial][clave] = diccionario[clave]
    return diccionario_por_inicial
```

Se carga el contenido del archivo txt en la variable *mi_texto* usando la función **lector_txt**:

[17]:
```
mi_texto = lector_txt("tu_me_quieres_blanca.txt")
```

Se obtiene un diccionario con la frecuencia de aparición de cada palabra en *mi_texto*, haciendo uso de la funcion **contador_palabras**:

[18]:
```
diccionario_frecuencias = contador_palabras(mi_texto)
```

Se pasa el diccionario obtenido a la función **alfabetizar_diccionarios**, para agrupar en diccionarios las palabras en base a su inicial:

[19]:
```
alfabetizar_diccionarios(diccionario_frecuencias)
```

[19]:
```
{'a': {'a': 3,
  'agua': 1,
  'al': 2,
  'alba': 4,
  'alcobas': 1,
  'alimenta': 1,
  'alma': 1,
  'amarga': 1,
```

```
 'azucena': 1},
'b': {'baco': 1,
 'banquete': 1,
 'bebe': 1,
 'blanca': 3,
 'boca': 1,
 'bosques': 1,
 'buen': 1},
'c': {'cabañas': 1,
 'carnes': 2,
 'casta': 3,
 'cerrada': 1,
 'con': 4,
 'conservas': 1,
 'copas': 1,
 'corola': 1,
 'corriste': 1,
 'cuando': 2,
 'cubierto': 1,
 'cuerpo': 1,
 'cuáles': 1},
'd': {'de': 8, 'dejaste': 1, 'del': 1, 'diga': 1, 'dios': 2, 'duerme': 1},
'e': {'el': 4,
 'ellas': 1,
 'en': 4,
 'engaño': 1,
 'enredada': 1,
 'entonces': 1,
 'escarcha': 1,
 'espumas': 1,
 'esqueleto': 1,
 'estrago': 1},
'f': {'festejando': 1, 'filtrado': 1, 'frutos': 1},
'h': {'habla': 1,
 'hacia': 1,
 'haya': 1,
 'hayas': 1,
 'hermana': 1,
 'hombre': 1,
 'hubiste': 1,
 'huye': 1},
'i': {'intacto': 1},
'j': {'jardines': 1},
'l': {'la': 3,
 'labios': 1,
 'las': 7,
 'lo': 2,
```

```
 'los': 4,
 'luna': 1,
 'lévate': 1,
 'límpiate': 1},
'm': {'mano': 1,
 'manos': 1,
 'margarita': 1,
 'me': 10,
 'mi': 1,
 'mieles': 1,
 'milagros': 1,
 'mojada': 1,
 'montaña': 1,
 'morados': 1},
'n': {'negros': 1, 'ni': 2, 'no': 1, 'nácar': 1, 'nívea': 2},
'p': {'perdone': 2,
 'perfume': 1,
 'por': 2,
 'pretendes': 3,
 'preténdeme': 3,
 'puesto': 1,
 'pájaros': 1,
 'pámpanos': 1},
'q': {'que': 6, 'quedó': 1, 'quieres': 6},
'r': {'rayo': 1, 'raíz': 1, 'renueva': 1, 'rocas': 1, 'rojo': 1},
's': {'salitre': 1, 'se': 2, 'sea': 1, 'sean': 1, 'sobre': 2, 'sé': 1},
't': {'te': 3,
 'tejidos': 1,
 'tenue': 1,
 'tierra': 1,
 'toca': 1,
 'todas': 2,
 'todavía': 1,
 'tornadas': 1,
 'tú': 8},
'u': {'un': 1, 'una': 1},
'v': {'vestido': 1, 'vete': 1, 'vive': 1},
'y': {'y': 5}}
```