IT ACADEMY – DATA ANALYTICS

Alumna: Josselyn Maritza Jumpa Gordillo

SPRINT 7

Resoldràs alguns problemes de la vida quotidiana aplicant les estructures de dades i control en Python.

Un client de l'empresa en la qual treballes demana una llista de programes molt senzills, però que li facilitarien molts processos. No obstant això, el departament de TU està molt complicat amb el temps, per la qual cosa et demanen que facis la programació.

NIVEL 1

EJERCICIO 1

Enunciado:

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. Pots obtenir més informació del seu càlcul en:
  - ➔ Índice de masa corporal IMC que es y como se calcula.
- La funció ha de classificar el resultat en les seves respectives categories

Respuesta:

Se han creado dos funciones. Una primera función "imc(peso,altura) que realiza el cálculo de índice de masa corporal, basándose en dos parámetros, la altura en metros y el peso en Kg. Esta función también clasifica el resultado en los rangos bajo peso, peso normal, sobrepeso y obesidad e imprime en pantalla la información de los parámetros, el índice de masa corporal, así como la clasificación obtenida.

Se muestra a continuación:

```
def imc (peso,altura):


        indice = peso / (altura **2)

        if indice < 18.5:
            rango = "bajo peso"
        elif 18.5 <= indice <= 24.9:
            rango = "peso normal"
        elif 25 <= indice <= 29.9:
            rango = "sobrepeso"
        else:
            rango = "obesidad"

        print(f"""
                Si su peso es {peso:.2f} Kg y su altura es {altura:.2f} m, su índice de masa corporal es {indice:.1f}.
                Se considera {rango}.""")
```

La segunda función “calculadora_imc()”, se utiliza para solicitar la información al usuario y asegurarnos de que se introducen dígitos y valores razonables para el peso o altura humanos. Una vez recibidos los inputs, esta función llama a la función que realiza el cálculo.

```
def calculadora_imc():

    print("""
            *** Bienvenidx a la calculadora de índice de masa corporal***

            Por favor, introduzca los datos sobre su peso y altura:

            """)

    while True:

        try:

            peso = float(input("Ingrese su peso en Kg. Por ejemplo: 70.5 :"))

            if peso <= 0 or peso > 400:
                print("Por favor, introduce un valor mayor a 0 Kg y no superior a 400 Kg ")

            else:

                while True:

                    try:

                        altura = float(input("Ingrese su altura en (m). Por ejemplo: 1.60 :"))

                        if altura <= 0 or altura > 3:
                            print("Por favor, introduce un valor mayor a 0 y no superior a 3 (m)")
                            continue


                        return imc(peso,altura)

                    except ValueError:

                        print("Por favor, introduce dígitos ")

        except ValueError:

            print("Por favor, introduce dígitos ")
```

El funcionamiento, sería el siguiente:

```
calculadora_imc()
```

```
            *** Bienvenidx a la calculadora de índice de masa corporal***

            Por favor, introduzca los datos sobre su peso y altura:


Ingrese su peso en Kg. Por ejemplo: 70.5 : 65

Ingrese su altura en (m). Por ejemplo: 1.60 : 1.62

Si su peso es 65.00 Kg y su altura es 1.62 m, su índice de masa corporal es 24.8.

Se considera peso normal.
```

Si el usuario introdujera un número por debajo o igual a 0, se mostraría un mensaje indicando que no es un valor válido y se volvería a solicitar introducir los datos:

```
Ingrese su peso en Kg. Por ejemplo: 70.5 : 0

Por favor, introduce un valor mayor a 0 Kg y no superior a 400 Kg
Ingrese su peso en Kg. Por ejemplo: 70.5 :
```

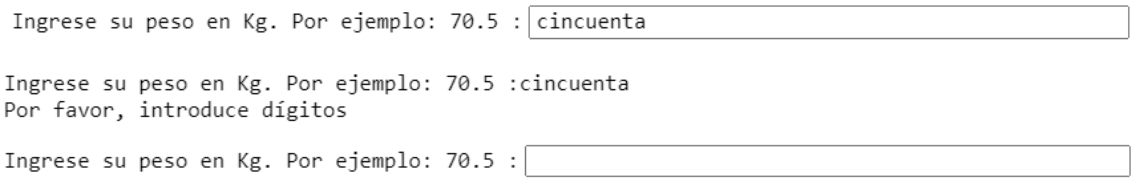
Lo mismo ocurre con la altura:

```
Ingrese su altura en (m). Por ejemplo: 1.60 : 0

Por favor, introduce un valor mayor a 0 y no superior a 3 (m)

Ingrese su altura en (m). Por ejemplo: 1.60 :
```

En el caso de que el usuario escribiera un valor no numérico, se le mostraría también un mensaje de error y volverían a solicitarse los datos:



EJERCICIO 2

Enunciado:

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 conversors, de tal manera que en introduir una temperatura retorni, com a mínim, dues conversions.

Respuesta:

Se han elegido las escalas de temperatura Celsius, Fahrenheit y Kelvin.

El usuario indicará en qué escala introduce la temperatura y la función mostrará la tabla de conversiones.

En esta ocasión se ha creado un total de 5 funciones.

3 funciones corresponden al cálculo de conversión de una escala a las otras dos escalas. Por ejemplo, “conversor_celsius(celsius)” convertirá una temperatura en centígrados que se ha pasado como parámetro, a su equivalente en Fahrenheit y Kelvin y mostrará los resultados en pantalla.

A continuación, se muestra el código de estas tres funciones de conversión:

```
def conversor_celsius(celsius):
    fahrenheit = celsius * (9/5) + 32
    kelvin = celsius + 273.15

    print(f"""
Celsius(ºC): {celsius:.2f}

Fahrenheit(ºF): {fahrenheit:.2f}

Kelvin (K): {kelvin:.2f}""")
```

```
def conversor_fahrenheit(fahrenheit):
    celsius = (fahrenheit - 32)/ (9/5)
    kelvin = celsius + 273.15

    print( f"""
Celsius(ºC): {celsius:.2f}

Fahrenheit(ºF): {fahrenheit:.2f}

Kelvin (K): {kelvin:.2f}""")
```

```
def conversor_kelvin(kelvin):
    celsius = kelvin - 273.15
    fahrenheit = (kelvin * (9/5)) - 459.7

    print( f"""
Celsius(ºC): {celsius:.2f}

Fahrenheit(ºF): {fahrenheit:.2f}

Kelvin (K): {kelvin:.2f}""")
```

Una vez definidas las funciones de cálculo, se ha creado otra función que acepta dos parámetros, el primero es la escala, representada por las letras ‘c’,’f’,’k’, y el segundo parámetro es una temperatura.

Según qué letra se pase como parámetro, la función indicará qué función de cálculo debe ejecutar. A continuación, se muestra el código:

```
def conversor(escala,temperatura):

    print("\nConversión de temperaturas:\n")


    if escala == 'c':
        return conversor_celsius(temperatura)
    elif escala == 'f':
        return conversor_fahrenheit(temperatura)
    else:
        return conversor_kelvin(temperatura)
```

La última función que se ha creado, presenta el programa, y solicita al usuario que introduzca valores para escala y temperatura.

Una vez se ha comprobado que los valores son válidos, se llama a la función conversor(escala,temperatura) que a su vez llama a la función correspondiente de cálculo de conversión.

A continuación, se muestra el código:

```
def conversor_temperaturas():
    print("""              ¡Bienvenidx al conversor de temperaturas de Celsius, Fahrenheit y Kelvin!

                                🟡❄️🌧️🌥️🟡❄️🌧️🌥️ 🟡❄️🌧️🌥️

                          Primero, indica qué en qué escala introducirás la temperatura:

                              "c" para Celsius
                              "f" para Fahrenheit
                              "k" para Kelvin

                          Luego, indica la temperatura.

                          Finalmente, obtendrás la conversión en las distintas escalas.


    """)
    escalas = ["c","k","f"]
    while True:

        try:

            escala = input("Escala: ").lower()
            escala = escala.strip()

            if escala not in escalas:

                print("Por favor, introduce 'c' para Celsius 'f' para Fahrenheit 'k' para Kelvin")


            else:

                while True:

                    try:

                        temperatura = float(input("Temperatura: "))

                        return conversor(escala,temperatura)

                    except ValueError:

                        print("Por favor, introduce dígitos")

        except ValueError:

            print("Por favor, introduce 'c' para Celsius 'f' para Fahrenheit 'k' para Kelvin")
```

El funcionamiento sería el siguiente:

```
conversor_temperaturas()
```

```
¡Bienvenidx al conversor de temperaturas de Celsius, Fahrenheit y Kelvin!

Primero, indica qué en qué escala introducirás la temperatura:

    "c" para Celsius
    "f" para Fahrenheit
    "k" para Kelvin

Luego, indica la temperatura.

Finalmente, obtendrás la conversión en las distintas escalas.

Escala: c
Temperatura: 30

Conversión de temperaturas:

Celsius(ºC): 30.00
Fahrenheit(ºF): 86.00
Kelvin(K): 303.15
```

En cuanto a la prevención de errores, si el usuario introdujese un valor no aceptado como escala, se le solicitaría nuevamente.

```
Escala: hola
Por favor, introduce 'c' para Celsius 'f' para Fahrenheit 'k' para Kelvin
Escala:
```

Igual ocurre con la temperatura, si se introdujera un valor erróneo, como por ejemplo una cadena de texto, el programa solicitaría nuevamente los datos:

```
Temperatura: treinta
Por favor, introduce dígitos
Temperatura:
```

EJERCICIO 3

Enunciado:

Comptador de paraules d'un text.

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

Respuesta:

En esta ocasión se han creado dos funciones, una para realizar la transformación del texto y recuento de palabras, y la otra función para solicitar al usuario que introduzca un texto y comprobar que se trata de una entrada válida y llamar a la primera función.

La primera función “contador_palabras(texto)”, recibe un texto como parámetro y realiza las siguientes transformaciones:

- Convierte todo a minúsculas para evitar distinciones
- Elimina los signos de puntuación.
  Para ello, se ha tomado como referencia los signos de puntuación que incluye la librería string “!"#$%&'()*+, -./:;<=>?@[\]^_`{|}~” y se han añadido dos caracteres más que se utilizan en español, como “¡¿”
- Separa la cadena de texto en una lista de palabras usando como separador los espacios en blanco.
- Elimina aquellos componentes de la lista que puedan no ser palabras y sólo espacios en blanco.
- A medida que recorre el listado de palabras con un bucle for, añade cada palabra a un diccionario como la clave y el recuento como valor.

Finalmente, imprime el texto original y el diccionario con cada palabra y su frecuencia de aparición.

A continuación, se muestra el código:

```
import string
def contador_palabras(texto):

    texto_lower = texto.lower()
    puntuacion = string.punctuation + "¡¿"
    texto_punctuation = texto_lower.translate(str.maketrans('', '', puntuacion))
    palabras_texto = texto_punctuation.split()
    palabras_frecuencia = {}



    for palabra in palabras_texto:

        if len(palabra)==0:
            palabras_texto.remove(palabra)


        palabras_frecuencia[palabra] = palabras_frecuencia.get(palabra,0) +1

    print(f"\n\nA continuación se muestra cuantas veces aparece cada palabra en el texto: \n\n",texto,"\n\n")
    print(palabras_frecuencia)
```

La segunda función, contador “contador_palabras_texto()” presenta brevemente el programa, solicita input al usuario, elimina los espacios en blanco de los extremos de la cadena con el método .strip() y verifica que el campo no haya quedado vacío. De lo contrario, vuelve a solicitarlo.

Luego, convierte ese input en el parámetro que pasará al llamar a la función “contador_palabras(texto)”.

A continuación, se muestra el código:

```python
def contador_palabras_texto():

    print("""
            ¡Bienvenidx al contador de palabras!

            Por favor, introduce un texto para obtener la frecuencia de aparición de cada palabra.

            ¡Importante!: No se hará distinción entre mayúsculas y minúsculas""")


    while True:

        texto = input("Por favor, introduce un texto: ").strip()

        if not texto:

             print("\nParece que el texto está vacío. Introduce un texto")

        else:

            return contador_palabras(texto)
```

El funcionamiento sería el siguiente:

```
            ¡Bienvenidx al contador de palabras!

            Por favor, introduce un texto para obtener la frecuencia de aparición de cada palabra.

            ¡Importante!: No se hará distinción entre mayúsculas y minúsculas
Por favor, introduce un texto: ¡Hola, hola! ¿Cómo estás?

A continuación se muestra cuantas veces aparece cada palabra en el texto:

 ¡Hola, hola! ¿Cómo estás?

{'hola': 2, 'cómo': 1, 'estás': 1}
```

En cuanto a la comprobación de errores, si el usuario dejara el campo vacío o introdujera únicamente espacios en blanco, volvería a solicitarse el input:

```
Por favor, introduce un texto:

Parece que el texto está vacío. Introduce un texto

Por favor, introduce un texto:
```

EJERCICIO 4

Enunciado:

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.

```python
reverse_dictionary({'a': 1, 'b': 2, 'c': 3})
```

```
{1: 'a', 2: 'b', 3: 'c'}
```

```
reverse_dictionary({'x': 'apple', 'y': 'banana', 'z': 'banana'})
```

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

Respuesta:

Se ha creado una función que recibe un diccionario como parámetro, itera por la clave,valor de todos sus ítems y añade cada valor a un diccionario nuevo, actuando como clave. La clave del diccionario original se añade como valor. Finalmente, imprime el nuevo diccionario

Este flujo de acciones sólo sucede si se comprueba que se trata de un elemento único no repetido ya en el nuevo diccionario que estamos creando. Si ya existiese una clave igual, se levanta un error que se muestra al usuario en pantalla y finaliza el flujo.

```
def reverse_dictionary(unsorted_dict):

    sorted_dictionary = {}

    try:

        for key,value in unsorted_dict.items():

            if value in sorted_dictionary.keys():
                raise ValueError("Error: multiple keys for one value")

            else:
                sorted_dictionary[value] = key


        return sorted_dictionary


    except ValueError as multiple_keys:

        print(multiple_keys)
```

A continuación, se muestra el funcionamiento en base a los ejemplos propuestos:

En este primer ejemplo, el diccionario que se pasa como parámetro contiene elementos no repetidos. Tras aplicar la función, vemos que las claves han pasado a ser los valores del nuevo diccionario y los valores se han convertido en las claves.

```
reverse_dictionary({'a':1, 'b':2,'c':3})
```

```
{1: 'a', 2: 'b', 3: 'c'}
```

En el caso de que uno de los elementos del diccionario parámetro sí esté repetido, se produce un error y el flujo finaliza.

```
reverse_dictionary({'x':'apple','y':'banana','z':'banana'})
```

```
Error, multiple keys for one value
```

NIVEL 2

EJERCICIO 1

Enunciado:

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'advertiment, 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.

```
reverse_dictionary_plus({'x': 'apple', 'y': 'banana', 'z': 'banana'})
{'apple': 'x', 'banana': ['y', 'z']}
```

Respuesta:

Retomando la estructura de la función “reverse_dictionary(unsorted_dict)”, se ha eliminado la condición que aplicábamos como excepción.

Al if statement que revisa si ya existe ese elemento en el nuevo diccionario, se ha programado que en caso de ser cierto, convierta los valores de ese item en una lista, aplicando la función list() . También se añadiría a esa lista la clave del diccionario original por la que se está iterando. Para esto último se ha utilizado el método append().

```
def reverse_dictionary_plus(unsorted_dict):

    sorted_dictionary = {}

    for key,value in unsorted_dict.items():

        if value in sorted_dictionary.keys():

            sorted_dictionary[value]=list(sorted_dictionary[value])
            sorted_dictionary[value].append(key)


        else:
            sorted_dictionary[value] = key


    return sorted_dictionary
```

A continuación, se muestra el funcionamiento en base al ejemplo propuesto:

Del diccionario introducido como parámetro se extrae el valor y se convierte en la clave del nuevo diccionario. A su vez, la clave del diccionario original se convierte en el valor del nuevo diccionario.

En el caso de que uno de los valores del parámetro aparezca más de una vez, sus claves conformarán una lista de valores en el diccionario resultante.

```
reverse_dictionary_plus({'x':'apple','y':'banana','z':'banana'})
{'apple': 'x', 'banana': ['y', 'z']}
```

EJERCICIO 2

Enunciado:

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']]

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

```
([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'])
```

Respuesta:

Se ha creado una función que recibe un parámetro de tipo lista. Dentro de la función se han creado dos listas vacías, una designada para los elementos que se convertirán en datatype float y otra lista para los elementos que no puedan convertirse.

La función comienza a iterar por cada ítem del parámetro recibido e intenta convertirlo en float mediante la función float() y añadirlo a la lista designada con el método .append().

Si el tipo de dato del ítem por el que se está iterando no es compatible con float() se recibirá un error, esto ocurre, por ejemplo con las tuplas que pueda haber dentro de la lista que recibimos como parámetro.

Para esa situación mediante except TypeError, se empezará a iterar por cada elemento dentro del ítem y se intentará nuevamente la conversión y adhesión a la lista de floats.

Si el tipo de valor no es convertible, mediante except ValueError, ese elemento se añade a la lista de no convertidos.

También utilizaremos exceptValueError en el primer bucle de iteración para añadir aquellos elementos que sí son compatibles con la función float() pero que no pueden convertirse en un valor numérico.

Finalmente, se imprimen las dos listas resultantes, aquellos valores convertidos a float y aquellos que no pudieron convertirse.

Se ha considerado imprimir en pantalla las listas resultantes sólo cuando contienen elementos, para evitar imprimir una lista vacía. Sin embargo, debido a que el enunciado indicaba que debían generarse dos listas, se ha optado por imprimir ambas en cualquier caso.

A continuación se muestra el código:

```python
def conversor_data_type(lista):

    converted_float = []
    not_converted = []

    for item in lista:

        try:

            converted_float.append(float(item))

        except TypeError:

            try:

                for element in item:

                    converted_float.append(float(element))

            except ValueError:

                not_converted.append(element)

        except ValueError:

            not_converted.append(item)


    print(converted_float)
    print(not_converted)
```

A continuación, se muestra el funcionamiento en base al ejemplo propuesto:

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

```
[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']
```

Observamos que también se itera y logra convertir alguno de los elementos dentro de la lista o tupla incluida en la lista principal que se pasa como parámetro.

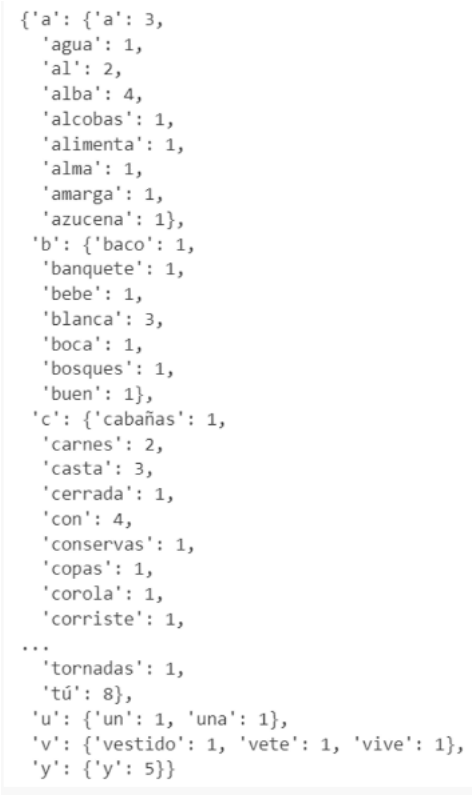
NIVEL 3

EJERCICIO 1

Enunciado:

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:



Respuesta:

En esta ocasión también se han creado dos funciones.

Una primera función “contar_ordenar_palabras(documento)” que acepta como parámetro una cadena de texto. Esta función sigue en gran medida la estructura del contador de palabras que creamos anteriormente.

Así, convierte todos los caracteres a minúsculas, elimina los signos de puntuación y mediante el método.split() crea una lista de palabras.

Es en este paso cuando se ordena alfabéticamente la lista empleando el método .sort(). También creamos un diccionario vacío que iremos llenando en los siguientes pasos.

Luego, a través de un bucle for se itera por cada elemento de la lista y se eliminan aquellos que puedan ser sólo espacios y así aseguramos que la lista sólo incluye palabras.

En ese mismo bucle, extraemos la primera letra de cada palabra y si no está incluida como clave en nuestro diccionario, la añadimos y como valor asignamos un diccionario vacío. De este modo, empezamos a crear el diccionario anidado. Para este proceso hemos utilizado slicing y las keyword not in y el método .keys().

Continuando en ese mismo bucle y nuevamente utilizando slicing nos adentramos en el diccionario anidado de cada letra y añadimos la palabra en su letra correspondiente y un recuento de sus apariciones. Para este proceso, hemos utilizado el método .get(), de la misma manera que se empleó para la función contador de palabras.

Finalmente, se imprime en pantalla el diccionario creado.

A continuación, se muestra el código:

```
def contar_ordenar_palabras(documento):


    texto_lower = documento.lower()
    puntuacion = string.punctuation + "¡¿"
    texto_punctuation = texto_lower.translate(str.maketrans('', '', puntuacion))
    palabras_texto = texto_punctuation.split()
    palabras_texto.sort()
    palabras_frecuencia = {}


    for palabra in palabras_texto:

        if len(palabra)==0:
            palabras_texto.remove(palabra)

        if palabra[0] not in palabras_frecuencia.keys():
            palabras_frecuencia[palabra[0]] = {}

        palabras_frecuencia[palabra[0]][palabra] = palabras_frecuencia[palabra[0]].get(palabra,0) +1


    print(f"\n\nA continuación, se muestra cuantas veces aparece cada palabra en el texto: \n\n")
    print(palabras_frecuencia)
```

La segunda función “palabras_documento”(file_path) acepta un filepath como parámetro, abre el archivo, lee el documento que incluye y llama a la función “contar_ordenar_palabras(documento)” y pasa el documento como parámetro.

Para este proceso se ha utilizado la estructura with open() as para asegurarnos que el archivo se abre únicamente para desarrollar el flujo de acciones y luego vuelve a cerrarse.

Para obtener una lectura correcta de los caracteres, se ha aplicado encoding ‘utf-8’.

Para esta función hemos considerado dos situaciones de error que pudieran surgir al leer los archivos, por lo tanto se ha utilizado una estructura try-except con los errores FileNotFoundError y OSError.

A continuación, se muestra el código:

```
def palabras_documento(file_path):


    try:
        with open(file_path,'r',encoding='utf-8') as documento_open:

            documento = documento_open.read()

            contar_ordenar_palabras(documento)

    except FileNotFoundError:

        print(f'''
                La ruta del archivo no pudo abrirse. Por favor, comprueba que la ruta sea correcta.
                Por ejemplo: r"C:\Documents\blanca.txt"''')

    except OSError as open_error:

        print(f'''
                La ruta del archivo no pudo abrirse. Por favor, comprueba que la ruta sea correcta
                Por ejemplo: r"C:\Documents\blanca.txt"''')
```

A continuación, se muestra el funcionamiento en base al ejemplo propuesto:

```
palabras_documento(r"C:\Users\moich\Documents\IT Academy _ Data analytics\7 Sprint\Tasca S7.01. Estructuras de datos y de control\tu_me_quieres_blanca.txt")
```

```
A continuación, se muestra cuantas veces aparece cada palabra en el texto:


{'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, 'ce
rrada': 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, 'límpiat
e': 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, 'prete
ndes': 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}}
```

Tal y como hemos explicado, las funciones reciben el archivo txt, realizan las transformaciones oportunas para separarlo en palabras y de ahí crea un diccionario principal con las letras del alfabeto encontradas en la primera letra de cada palabra. Luego, crea un diccionario anidado en cada una de esas entradas donde la clave será la palabra y el valor será el recuento de aparición.

En el caso de que la ruta del archivo se introdujera incorrectamente. (En este caso se ha eliminado la terminación .txt de la ruta), se recibiría el siguiente mensaje de error:

```
palabras_documento(r"C:\Users\moich\Documents\IT Academy _ Data analytics\7 Sprint\Tasca S7.01. Estructuras de datos y de control
```

```
La ruta del archivo no pudo abrirse. Por favor, comprueba que la ruta sea correcta
Por ejemplo: r"C:\Documentlanca.txt"
```