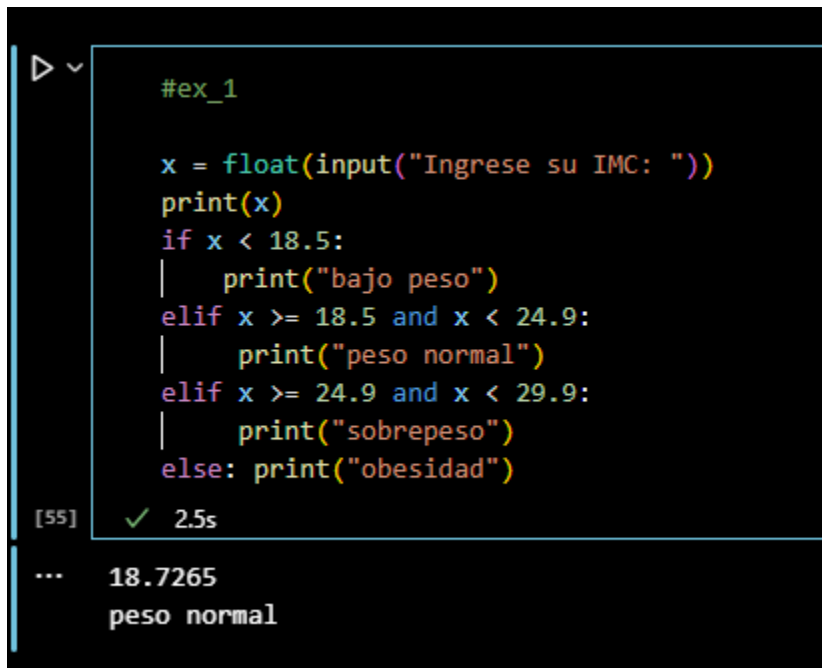


- Exercici 1

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. Pots obtenir més informació del seu càlcul en:
 - Índice de masa coporal IMC
 - La funció ha de classificar el resultat en les seves respectives categories



```
#ex_1

x = float(input("Ingrese su IMC: "))
print(x)
if x < 18.5:
    print("bajo peso")
elif x >= 18.5 and x < 24.9:
    print("peso normal")
elif x >= 24.9 and x < 29.9:
    print("sobrepeso")
else: print("obesidad")
```

[55] ✓ 2.5s

... 18.7265
peso normal

Esta sería la versión más sencilla, definiendo bien los rangos y trabajando con el input de ingresar el IMC. Con la ayuda de Chat GPT pude añadir los valores para hacer los cálculos, que sinceramente, era más sencillo de lo que me pensaba.

```
weight = float(input("Ingrese su peso en kg: "))
height = float(input("Ingrese su altura en cm: "))

# Convert height from cm to meters
height /= 100

# Calculate BMI
bmi = weight / (height ** 2)

# Determine BMI category
print(height)
print(weight)
print(bmi)
if bmi < 18.5:
    print("Bajo peso")
elif 18.5 <= bmi < 24.9:
    print("Peso normal")
elif 24.9 <= bmi < 29.9:
    print("Sobrepeso")
else:
    print("Obesidad")
```

[84] ✓ 2.5s

```
... 1.73
    78.0
    26.061679307694877
    Sobrepeso
```

De esta manera podemos añadir los dos valores, altura y peso, para que el usuario los introduzca y le haga el cálculo. Definimos las operaciones, altura/100 para tener la altura en metros y el cálculo del bmi. Imprimimos la altura y el peso para tener toda la información del usuario.

- 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.

Primero definimos las funciones que necesitamos para convertir la temperatura, en este caso, el valor que vamos a introducir (value), la unidad de temperatura original (input_scale) y a que unidad de temperatura la vamos a convertir (output_scale).

```
#ex_2
```

```
def convert_temperature(value, input_scale, output_scale):
```

```
    if input_scale == 'C':
        if output_scale == 'F':
            return value * 1.8 + 32
        elif output_scale == 'K':
            return value + 273.15
        elif output_scale == "N":
            return value / 3.03030303
        else:
            return value
    elif input_scale == 'F':
        if output_scale == 'C':
            return (value - 32) / 1.8
        elif output_scale == 'K':
            return (value + 459.67) * 5 / 9
        elif output_scale == "N":
            return (value - 32) * 11 / 60
        else:
            return value
    elif input_scale == 'K':
        if output_scale == 'C':
            return value - 273.15
        elif output_scale == 'F':
            return value * 9 / 5 - 459.67
        elif output_scale == "N":
            return (value - 273.15) * 33 / 100
        else:
            return value
    elif input_scale == "N":
        if output_scale == "C":
            return value * 3.03030303
        elif output_scale == "F":
            return (value * 60 / 11 + 32)
        elif output_scale == "K":
            return (value * 100 / 33 + 273.15)
        else:
            return value
    else:
        return value
```

Input_scale es la unidad de temperatura original, así que tenemos que definir la output_scale, temperatura a que se va a convertir y que cálculo se tiene que hacer para poder hacer la conversión. Lo hacemos con cada uno de los valores, Celsius, Fahrenheit, Kelvin y Newton.

```
while True:
    # Prompt the user for input
    print('Enter the input temperature value:')
    value = float(input())
    print('Enter the input temperature scale (C, F, K or N):')
    input_scale = input().upper()
    print('Enter the output temperature scale (C, F, K or N):')
    output_scale = input().upper()

    # Convert the temperature and print the result
    result = convert_temperature(value, input_scale, output_scale)
    print(f'{value} {input_scale} = {result} {output_scale}')

    # Prompt the user to continue or quit
    print('Enter q to quit, or any other key to continue:')
    choice = input()
    if choice.lower() == 'q':
        break
```

[28] ✓ 52s

```
... Enter the input temperature value:
Enter the input temperature scale (C, F, K or N):
Enter the output temperature scale (C, F, K or N):
20.0 C = 68.0 F
Enter q to quit, or any other key to continue:
```

Para terminar, hacemos que el loop se haga siempre y cuando se cumpla el orden de los factores. Número de la temperatura a convertir, unidad original, unidad a transformar, y si queremos parar el loop con la q o no. Usamos un float para asegurarnos que solo funcione con valores numéricos.

- Exercici 3

Comptador de paraules d'un text.

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

```
#ex 3

intro = input('Introduce aquí tu frase y contaremos las palabras:')
text = intro.split()
wordcount={}
freq = text.count(text)

for word in text:
    if word not in wordcount:
        wordcount[word] = 1
    else:
        wordcount[word] += 1

print(intro)

print('Este es tu recuento:')
for word, freq in wordcount.items():
    print(word, freq)
```

✓ 9.1s

hola juan me llamo pedro, hola pedro me llamo juan
Este es tu recuento:
hola 2
juan 2
me 2
llamo 2
pedro, 1
pedro 1

Este ejercicio lo he hecho de maneras diferentes, con y sin ayuda de chat gpt. Esta primera versión es muy sencilla de entender, pero no me separa los caracteres especiales, como las comas. Usamos el input para poder introducir la frase, separamos las palabras usando la función Split(), hacemos una lista usando wordcount{ } y contamos las veces que aparece cada palabra en text.count(text). Terminamos usando un loop para saber si hay que sumar palabras o no para poder contar la frecuencia. Imprimimos la frase, las palabras y la frecuencia.

Álvaro Míguez

Sprint 7: Python: Nocions I coneixements bàsics

```
#ex 3

from collections import Counter
intro = input('Introduce aquí tu frase y contaremos las palabras:')
text = intro.split()
wordcount = Counter(text)

print('Este es tu recuento:')
for word, freq in wordcount.items():
    print(word, freq)
```

✓ 6.5s

Este es tu recuento:
hola 2
juan 2
soy 2
pedro, 1
pedro 1

Aquí con la ayuda de Chat GPT me recomienda usar la librería collections para poder contar las palabras. Como se define en la web <https://docs.python.org/3/library/collections.html> , counter - dict subclass for counting [hashable](#) objects

```
# Import the re module
import re
# string with special characters
special_string="I did what I did and I wish I hadn't"
print("String before conversion: ",special_string)
#using regular expression with the sub() method
normal_string =re.sub("[^A-Z]", " ", special_string,0,re.IGNORECASE)
print("String after conversion: ",normal_string)
text = special_string.split()
for word in text:
    freq = text.count(word)
    print(word, freq)
```

✓ 0.0s

String before conversion: I did what I did and I wish I hadn't
String after conversion: I did what I did and I wish I hadn t
I 4
did 2
what 1
I 4
did 2
and 1
I 4
wish 1
I 4
hadn't 1

Aquí importamos el módulo RE que según w3

(https://www.w3schools.com/python/python_regex.asp) sirve para contar expresiones regulares. Lo divertido de este código es que por un lado imprimimos caracteres especiales (special_string) y luego lo normaliza para quitar en el caso del ejemplo, el apostrofe en hadn't. Lo que no me gusta es que me repite las letras en el conteo de frecuencias. Tengo una última versión que dejo en el archivo.

Corrección Eduard Carnero:

Eduard me recomendó crear un diccionario y jugar con la frecuencia (freq) fuera del loop. Al final después de trastear un poco lo que funcionó fue crear el diccionario Wordcount y cambiar el loop para que contase de manera simple la frecuencia de las palabras. Queda de la siguiente manera

```
#ex3 (dicts y conversion normalizadas)

# Import the re module
import re
# string with special characters
special_string="I did what I did, and I wish I hadn't"
print("String before conversion: ",special_string)
#using regular expression with the sub() method
normal_string =re.sub("[^A-Z]", " ", special_string,0,re.IGNORECASE)
print("String after conversion: ",normal_string)
text = special_string.split()
wordcount={}

for word in text:
    freq = text.count(word)
    #opcion de crear un diccionario dentro del loop.
    if word not in wordcount:
        wordcount[word] = 1
    else:
        wordcount[word] += 1
print(wordcount)
```

```
String before conversion: I did what I did, and I wish I hadn't
String after conversion: I did what I did and I wish I hadn t
{'I': 4, 'did': 1, 'what': 1, 'did.': 1, 'and': 1, 'wish': 1, 'hadn't': 1}
```

- 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.

```
my_map = {'i': 1, 'like': 1, 'pot@toes': 1}

res = dict(reversed(list(my_map.items())))

print(str(res))
```

✓ 0.0s

```
{'pot@toes': 1, 'like': 1, 'i': 1}
```

Mi primera versión fue muy sencilla y no funcionaba, solo hacia el reverso de los ítems entregados. Esta segunda versión sale de este enlace <https://www.geeksforgeeks.org/python-find-keys-with-duplicate-values-in-dictionary/>

```
#ex4
my_map = {'i': 1, 'like': 1, 'pot@toes': 1}
flipped = {}
for key, value in my_map.items():
    if value not in flipped:
        flipped[value] = [key]
        print(flipped)
    else:
        raise KeyError("duplicates exist")
```

⊗ 0.0s

```
{1: ['i']}
```

KeyError Traceback (most recent call last)

Cell In[98], line 9

```
7 print(flipped)
8 else:
--> 9 raise KeyError("duplicates exist")
```

KeyError: 'duplicates exist'

Aquí conseguimos avanzar, el error salta, pero, aun así, me da una respuesta de valores invertidos en la lista. Lo importante aquí es que entendemos que los valores invertidos se pueden eliminar si se repiten.

```
#ex 4(works)
my_map = {'i': 1, 'like': 2, 'pot@toes':3}
res = dict((v, k) for k, v in my_map.items())
if len(my_map) != len(set(res)):
    raise ValueError("Existen duplicados")
else: print(res)

✓ 0.0s

{1: 'i', 2: 'like', 3: 'pot@toes'}
```

Esta versión conseguimos que funcione y realmente es un código muy sencillo a la que entendemos que los valores duplicados una vez invertidos se pueden eliminar. Lo único que le pedimos es que si la longitud (len) de los ítems en my_map no son los mismos cuando se hace la lista invertida (res) entonces me tiene que dar el error. Gracias a eso podemos tener el resultado correcto tanto con y sin duplicados.

```
#ex 4(works)
my_map = {'i': 1, 'like': 1, 'pot@toes':1}
res = dict((v, k) for k, v in my_map.items())
if len(my_map) != len(set(res)):
    raise ValueError("Existen duplicados")
else: print(res)

⊗ 0.0s

-----
ValueError                                Traceback (most recent call last)
Cell In[94], line 5
      3 res = dict((v, k) for k, v in my_map.items())
      4 if len(my_map) != len(set(res)):
----> 5     raise ValueError("Existen duplicados")
      6 else: print(res)

ValueError: Existen duplicados
```

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 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.

```
#ex2.1
my_map = {'i': 1, 'like': 1, 'pot@toes': 1}
e = {}

for k, v in my_map.items():
    e.setdefault(v, set()).add(k)

print(e)
```

```
{1: {'like', 'i', 'pot@toes'}}
```

Este código lo saqué de este enlace: <https://stackoverflow.com/questions/73221511/how-to-reverse-a-dictionary-with-repeated-values-as-a-set>

Donde lo que hacemos es que me responda e, que es un diccionario. Lo que le pedimos es que me coja las keys y values de my_map y que para cada value concreto, en este caso 1, me dé su key que lo acompaña, en caso de que no se repita me tiene que añadir una nueva key. Al repetirse me crea el set() donde todos los values al ser 1 me va añadiendo sus keys.

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']]

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']]

```
#ex 2.2

llista = [ '1.3', 'one' , '1e10' , 'seven', '3-1/2', ('2',1,1.4,'not-a-number'), [1,2,'3','3.4'] ]
zip(tuple([1,2,'3','3.4']))
for item in llista:
    try:
        numeric_part = ''.join(char for char in llista if char.isdigit() or char == '.')
        cleaned_string = llista.replace(',', '')
        trimmed_string = llista.strip()
        result = float(numeric_part, cleaned_string, trimmed_string)
        print(result)
    except AttributeError:
        llista.remove(item)
print(llista)
```

✓ 0.0s

['one', 'seven', ('2', 1, 1.4, 'not-a-number')]

Utilicé este código para poder hacer las primeras separaciones, lo que sucede es que,

- a) me elimina todos los floats
- b) descubrí más adelante que el último item es un tuple.

Así que no termina de funcionar pese a limpiar la lista del cliente.

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

floats = []
non_floats = []

for i in lista:
    if isinstance(i, tuple): # Check if the item is a tuple
        tuple_floats = []
        tuple_non_floats = []
        for sub_item in i:
            try:
                if isinstance(sub_item, (int, float)): # Check if the sub-item is already a float
                    tuple_floats.append(sub_item)
                else:
                    float_value = float(str(sub_item))
                    tuple_floats.append(float_value)
            except ValueError:
                tuple_non_floats.append(sub_item)
        floats.extend(tuple_floats)
        non_floats.extend(tuple_non_floats)
    else: # If the item is not a tuple
        try:
            float_value = float(str(i))
            floats.append(float_value)
        except ValueError:
            non_floats.append(i)

print("Floats:", floats)
print("Non-floats:", non_floats)
```

```
Floats: [1.3, 10000000000.0, 2.0, 1, 1.4, 1, 2, 3.0, 3.4]
Non-floats: ['one', 'seven', '3-1/2', 'not-a-number']
```

Con la ayuda de chat gpt conseguimos sacarlo de esta manera. Primero creamos las dos listas de floats y non_floats. Usamos un isinstance para chequear que los ítems del tuple son floats o no son floats y que cada sub_item, ítem dentro del tuple, se añada a la lista de floats o non_floats si cumplen con la condición, en caso de no cumplir con la condición de ser un float y salta el ValueError entonces se añade a la lista de non_floats. Se hace lo mismo con el resto de ítems fuera del tuple.

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

floats = []
non_floats = []

for item in lista:
    if isinstance(item, (int, float)):
        floats.append(item)
    elif isinstance(item, str):
        try:
            float_value = float(item)
            floats.append(float_value)
        except ValueError:
            non_floats.append(item)
    elif isinstance(item, tuple):
        for sub_item in item:
            if isinstance(sub_item, (int, float)):
                floats.append(sub_item)
            elif isinstance(sub_item, str):
                try:
                    float_value = float(sub_item)
                    floats.append(float_value)
                except ValueError:
                    non_floats.append(sub_item)

print("Floats:", floats)
print("Non-floats:", non_floats)
```

```
Floats: [1.3, 10000000000.0, 2.0, 1, 1.4, 1, 2, 3.0, 3.4]
Non-floats: ['one', 'seven', '3-1/2', 'not-a-number']
```

Con un poco de juego podemos hacer que el chequeo de los ítems dentro del tuple estén mejor integrados en el código.

Corrección Edu Carnero:

Edu me motivó a intentar simplificar el código, lo que conseguí que funcionase es separarlo por tipos de ítem (str, int, tuple) y tratarlos por igual tanto los ítems como los sub_ítem. De manera que trabajamos tanto dentro como fuera del tuple por igual.

```
#codigo simplificado:
```

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

```
floats = []  
non_floats = []
```

```
for item in lista:  
    if isinstance(item, (int, float)):  
        floats.append(item)  
    elif isinstance(item, str):  
        try:  
            float_value = float(item)  
            floats.append(float_value)  
        except ValueError:  
            non_floats.append(item)  
    elif isinstance(item, tuple):  
        for sub_item in item:  
            try:  
                if isinstance(sub_item, (int, float)):  
                    floats.append(sub_item)  
            else:  
                float_value = float(str(sub_item))  
                floats.append(float_value)  
        except ValueError:  
            non_floats.append(sub_item)
```

```
print("Floats:", floats)  
print("Non-floats:", non_floats)
```

```
Floats: [1.3, 10000000000.0, 2.0, 1, 1.4, 1, 2, 3.0, 3.4]  
Non-floats: ['one', 'seven', '3-1/2', 'not-a-number']
```

Nivel 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".

```
# Open the file
with open('D:\\tu_me_quieres_blanca.txt', 'r') as file:
    # Read the file content
    content = file.read()

# Split the content into words
words = content.split()

# Create a dictionary to store word frequencies
word_freq = {}

# Count word frequencies
for word in words:
    word = word.lower() # Convert word to lowercase to ensure case-insensitive counting
    word_freq[word] = word_freq.get(word, 0) + 1

# Sort the dictionary by keys (words) alphabetically
sorted_word_freq = sorted(word_freq.items())

# Print the sorted word frequencies
for word, freq in sorted_word_freq:
    print(f'{word}: {freq}')
```

✓ 0.0s

Con la ayuda de chat gpt este fue el primer código con el que trasteo. Si que separa, ordena y cuenta la frecuencia de las palabras, pero no respeta los caracteres especiales, como por ejemplo en las respuestas de abajo cabañas se lee mal. Mandamos al código a leer el archivo, separamos las palabras y creamos un diccionario para la frecuencia, la damos la orden de sumar la frecuencia (Word_freq.get(Word,0) +1) y después lo ordenamos (sorted)

```

al: 2
alba!: 1
alba,: 1
alba.: 2
alcobas: 1
alimenta: 1
alma: 1
amarga,: 1
azucena: 1
baco.: 1
banquete: 1
bebe: 1
blanca: 1
blanca,: 2
boca,: 1
bosques,: 1
buen: 1
cabañas,: 1
carnes: 2
casta: 1
casta.: 2

```

```

import re

# Open the file
with open('D:\\tu_me_quieres_blanca.txt', 'r') as file:
    # Read the file content
    content = file.read()

# Extract words using regular expressions
words = re.findall(r'\b[a-zA-ZÑñÁáÉéÍíÓóÚúÜü.,;'\\""]\b', content.lower(), flags=re.UNICODE)

# Create a dictionary to store word frequencies
word_freq = {}

# Count word frequencies and group them by starting letter
for word in words:
    first_letter = word[0]
    word_freq.setdefault(first_letter, {})
    word_freq[first_letter][word] = word_freq[first_letter].get(word, 0) + 1

# Sort the dictionary by keys (letters) alphabetically
sorted_word_freq = sorted(word_freq.items())

# Print the sorted word frequencies with titles for each letter
for letter, words_dict in sorted_word_freq:
    print(f'\nWords starting with letter {letter.upper()}:')
    for word, freq in sorted(words_dict.items()):
        print(f'{word}: {freq}')

```

✓ 0.0s

Álvaro Míguez

Sprint 7: Python: Nocions I coneixements bàsics

Jugando un poco más con chatgpt importamos re y tenemos que jugar con el regex para poder incluir los acentos, ñ y puntuaciones. Añadiendo first_letter podemos ver cuándo empieza la siguiente letra en la clasificación.

```
Words starting with letter A:
```

```
a: 5
```

```
agua: 1
```

```
al: 2
```

```
alba: 4
```

```
alcobas: 1
```

```
alimenta: 1
```

```
alma: 1
```

```
amarga: 1
```

```
as: 1
```

```
azucena: 1
```

```
Words starting with letter B:
```

```
baco: 1
```

```
banquete: 1
```

```
bebe: 1
```

```
blanca: 3
```

```
boca: 1
```

```
bosques: 1
```

```
buen: 1
```

```
#ex 3.1 (works)

import re

# Set the file path
file_path = "D:\\tu_me_quieres_blanca.txt"

# Read the text from the file
with open(file_path, 'r', encoding='utf-8') as file:
    text = file.read()

# Extract words using regular expressions
words = re.findall(r'\b[a-zA-ZñÑáéíóúÁÉÍÓÚÜ.,;\'\\"'[-]+\b', text)

# Count word frequencies
wordfreq = {}
for word in words:
    word = word.lower()
    wordfreq[word] = wordfreq.get(word, 0) + 1

# Sort the dictionary by keys (words) alphabetically
sorted_wordfreq = sorted(wordfreq.items())

prev_first_letter = None

print("Words ordered alphabetically with their frequencies:")

for word, freq in sorted_wordfreq:
    first_letter = word[0]
    if first_letter != prev_first_letter:
        print({first_letter.upper()})
        prev_first_letter = first_letter
    print(f'{word}: {freq}')
```

✓ 0.0s

Por último, con este código al principio tuve problemas por que me normalizaba el código y eso hacía que cabañas y acentos no salieran correctamente, tuve que eliminar esa parte del código para poder hacer que funcionase como debería. Junto a eso añadimos la línea de código de: “if first_letter != prev_first_letter: print({first_letter.upper()})” para poder separar las palabras por letra inicial.

```
Words ordered alphabetically with their frequencies:
{'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
...
vete: 1
vive: 1
{'Y'}
y: 5
```

Álvaro Míguez

Sprint 7: Python: Nocions I coneixements bàsics

Corrección Edu Carnero:

Edu me dijo que no estaba cumpliendo debidamente con el ejercicio ya que estaba creando una lista y haciendo su print en lugar de crear bien un diccionario (nota: puede que aquí la memoria me falle).

En su lugar he conseguido que el 2º código que muestro funcione debidamente.

```
import re

# Open the file
with open('D:\\tu_me_quieres_blanca.txt', 'r', encoding='utf-8') as file:
    # Read the file content
    content = file.read()

# Extract words using regular expressions
words = re.findall(r'\b[a-zA-ZñÁáÍíÚúÁÁÉÉÍÍÓóÚú.,;'\\"'[]\-\]+\b', content.lower(), flags=re.UNICODE)

# Create a dictionary to store word frequencies
word_freq = {}

# Count word frequencies and group them by starting letter
for word in words:
    first_letter = word[0]
    word_freq.setdefault(first_letter, {})
    word_freq[first_letter][word] = word_freq[first_letter].get(word, 0) + 1

# Sort the dictionary by keys (letters) alphabetically
sorted_word_freq = sorted(word_freq.items())

# Print the sorted word frequencies with titles for each letter
for letter, words_dict in sorted_word_freq:
    print(f'\nWords starting with letter {letter.upper()}:')
    for word, freq in sorted(words_dict.items()):
        print(f'{word}: {freq}')
```

Nota curiosa, una cosa que me funciona para que se vean los caracteres especiales españoles más allá del regex es usar “encoding = utf-8”

```
Words starting with letter A:
a: 3
agua: 1
al: 2
alba: 4
alcobas: 1
alimenta: 1
alma: 1
amarga: 1
azucena: 1

Words starting with letter B:
baco: 1
banquete: 1
bebe: 1
blanca: 3
boca: 1
bosques: 1
buen: 1

Words starting with letter C:
cabañas: 1
carnes: 2
casta: 3
...
vive: 1

Words starting with letter Y:
y: 5
```