Desafío 1: Tuplas

¿Sabías que puedes crear tuplas con solo un elemento?

En la celda de abajo, define una variable `tup` con un único elemento `"I"`.

```
tup = tuple("I")
```

Imprime el tipo de `tup`.

Asegúrate de que su tipo sea correcto (es decir, *tuple* en lugar de *str*).

```
type(tup)
tuple
print(tup)
('I',)
```

Ahora intenta agregar los siguientes elementos a `tup`.

¿Puedes hacerlo? Explica.

```
"r", "o", "n", "h", "a", "c", "k',
# No, Las tuplas son objetos inmutables
```

¿Qué tal si reasignas un nuevo valor a una tupla existente?

Reasigna los siguientes elementos a `tup`. ¿Puedes hacerlo? Explica.

```
"I", "r", "o", "n", "h", "a", "c", "k"
tup=tuple("Ironhack")
print(tup)
('I', 'r', 'o', 'n', 'h', 'a', 'c', 'k')
```

Divide `tup` en `tup1` y `tup2` con 4 elementos en cada una.

`tup1` debe ser `("I", "r", "o", "n")` y `tup2` debe ser `("h", "a", "c", "k")`.

Sugerencia: usa números de índice positivos para la asignación de `tup1` y números de índice negativos para la asignación de `tup2`. Los números de índice positivos cuentan desde el principio, mientras que los números de índice negativos cuentan desde el final de la secuencia.

También imprime `tup1` y `tup2`.

```
tup1=tup[0:4]
tup2=tup[-4:]

print(f"tup1: {tup1}")
print(f"tup2: {tup2}")

tup1: ('I', 'r', 'o', 'n')
tup2: ('h', 'a', 'c', 'k')
```

Suma `tup1` y `tup2` en `tup3` usando el operador `+`.

Luego imprime `tup3` y verifica si `tup3` es igual a `tup`.

```
tup3=tup1+tup2

print(tup3)

('I', 'r', 'o', 'n', 'h', 'a', 'c', 'k')

print(tup3 == tup)

True

print(tup3 is tup)

False
```

Cuenta el número de elementos en `tup1` y `tup2`. Luego suma los dos conteos y verifica si la suma es igual al número de elementos en `tup3`.

```
n_tup1=tup1.__len__()
n_tup2=tup2.__len__()
n_tup3=tup3.__len__()

print(n_tup3 == (n_tup1+n_tup2))

True
```

¿Cuál es el número de índice de `"h"` en `tup3`?

```
tup3.index("h")

4
```

Ahora, usa un bucle FOR para verificar si cada letra en la siguiente lista está presente en `tup3`:

```
letters = ["a", "b", "c", "d", "e"]
```

Para cada letra que verifiques, imprime `True` si está presente en `tup3`, de lo contrario imprime `False`.

Sugerencia: solo necesitas hacer un bucle con `letters`. No necesitas hacer un bucle con `tup3` porque hay un operador de Python `in` que puedes usar. Consulta la referencia.

```
letters = ["a", "b", "c", "d", "e"]

x=range(0,11) #[0,1,2,3,4,5,6,7,8,9,10]

for sdfdsfs in letters:
    print(sdfdsfs)
```

```
a
b
c
d
e
```

```
num = letters[0]
num = letters[1]
num = letters[2]
num = letters[3]
num = letters[4]

for i in letters:
    if(i in tup3):
        print(f"TRUE, La letra {i} se encuentra en tup3")
    else:
        print(f"FALSE, La letra {i} no se encuentra en tup3")
```

```
TRUE, La letra a se encuentra en tup3
FALSE, La letra b no se encuentra en tup3
TRUE, La letra c se encuentra en tup3
FALSE, La letra d no se encuentra en tup3
FALSE, La letra e no se encuentra en tup3
```

¿Cuántas veces aparece cada letra de `letters` en `tup3`?

Imprime el número de ocurrencias de cada letra.

```
for i in letters:
    if(i in tup3):
        print(f"La letra {i} tiene {tup3.count(i)} ocurrencia(s) en
tup3")
    else:
        print(f"La letra {i} tiene 0 ocurrencias en tup3")
```

```
La letra a tiene 1 ocurrencia(s) en tup3
La letra b tiene 0 ocurrencias en tup3
La letra c tiene 1 ocurrencia(s) en tup3
La letra d tiene 0 ocurrencias en tup3
La letra e tiene 0 ocurrencias en tup3
```

Desafío 2: Conjuntos

Hay mucho que aprender sobre los Conjuntos en Python y la información presentada en la lección es limitada debido a su longitud. Para aprender a fondo sobre los Conjuntos en Python, te recomendamos encarecidamente que revises el tutorial de W3Schools sobre Ejemplos y Métodos de Conjuntos en Python antes de trabajar en este laboratorio. Algunas preguntas difíciles de este laboratorio tienen sus soluciones en el tutorial de W3Schools.

Primero, importa la biblioteca `random` de Python.

```
import random
```

En la celda de abajo, crea una lista llamada `sample_list_1` con 80 valores aleatorios.

Requisitos:

- Cada valor es un entero entre 0 y 100.
- Cada valor en la lista es único.

Imprime `sample_list_1` para revisar sus valores.

Sugerencia: usa `random.sample` (referencia).

```
sample_list_1=random.sample(range(101),k=80);
len(sample_list_1)
```

```
80
```

Convierte `sample_list_1` en un conjunto llamado `set1`. Imprime la longitud del conjunto. ¿Su longitud sigue siendo 80?

```
set1=set(sample_list_1)
len(set1)
```

```
80
```

Crea otra lista llamada `sample_list_2` con 80 valores aleatorios.

Requisitos:

- Cada valor es un entero entre 0 y 100.
- Los valores en la lista no tienen que ser únicos.

Sugerencia: Usa un bucle FOR.

```
sample_list_2=[]
for i in range(0,80):
    sample_list_2.extend(random.sample(range(0,101),k=1))
len(sample_list_2)
```

```
80
```

Convierte `sample_list_2` en un conjunto llamado `set2`. Imprime la longitud del conjunto. ¿Su longitud sigue siendo 80?

```
set2=set(sample_list_2)
len(set2)
```

```
59
```

Identifica los elementos presentes en `set1` pero no en `set2`. Asigna los elementos a un nuevo conjunto llamado `set3`.

```
set3=set1.difference(set2)
len(set3)
```

```
33
```

Identifica los elementos presentes en `set2` pero no en `set1`. Asigna los elementos a un nuevo conjunto llamado `set4`.

```
set4=set2.difference(set1)
len(set4)
```

```
12
```

Ahora identifica los elementos compartidos entre `set1` y `set2`. Asigna los elementos a un nuevo conjunto llamado `set5`.

```
set5=set1&set2
len(set5)
```

```
47
```

¿Cuál es la relación entre los siguientes valores?

- len(set1)
- len(set2)
- len(set3)
- len(set4)
- len(set5)

Usa una fórmula matemática para representar esa relación. Prueba tu fórmula con código Python.

```
len(set1)>=len(set2)
```

```
True
```

```
len(set3)==(len(set1)-len(set5))
```

```
True
```

```
len(set4)==(len(set2)-len(set5))
```

```
True
```

```
len(set5)<=len(set1)
```

```
True
```

Crea un conjunto vacío llamado `set6`.

```
set6=set()
len(set6)
```

```
0
```

Añade `set3` y `set5` a `set6` usando el método `update` de los Conjuntos de Python.

```
set6.update(set3)
set6.update(set5)
len(set6)
```

```
80
```

Verifica si `set1` y `set6` son iguales.

```
print(set1==set6)
```

```
True
```

```
print(set1 is set6)
```

```
False
```

Comprueba si `set1` contiene a `set2` utilizando el método `issubset` de los Conjuntos de Python. Luego verifica si `set1` contiene a `set3`.

```
set2.issubset(set1)
```

```
False
```

```
set3.issubset(set1)
```

```
True
```

Utilizando el método `union` de los Conjuntos de Python, agrega `set3`, `set4` y `set5`. Luego agrega `set1` y `set2`.

Verifica si los valores agregados son iguales.

```
set7=set()
set7=set3|set4|set5
len(set7)
```

```
92
```

```
set8=set()
set8=set1|set2
len(set8)
```

```
92
```

```
print(set7 == set8)
```

```
True
```

```
print(set7 is set8)
```

```
False
```

Utilizando el método `pop`, elimina el primer elemento de `set1`.

```
set9=set1.copy()
len(set9)
```

```
80
```

```
print(set9 == set1)
```

```
True
```

```
set9.pop()
len(set9)
```

```
79
```

```
print(set9 == set1)
```

```
False
```

Elimina cada elemento en la siguiente lista de `set1` si están presentes en el conjunto. Imprime los elementos restantes.

```
list_to_remove = [1, 9, 11, 19, 21, 29, 31, 39, 41, 49, 51, 59, 61,
69, 71, 79, 81, 89, 91, 99]
```

```
list_to_remove = [1, 9, 11, 19, 21, 29, 31, 39, 41, 49, 51, 59, 61,
69, 71, 79, 81, 89, 91, 99]

len(set1)
```

```
80
```

```
for i in list_to_remove:
    if i in set1:
        set1.remove(i)
len(set1)
```

```
64
```

BONUS - Desafío 3: Diccionarios

En este desafío practicarás cómo manipular diccionarios en Python. Antes de empezar este desafío, te animamos a revisar los Ejemplos y Métodos de Diccionarios en Python de W3School.

Lo primero que practicarás es cómo ordenar las claves en un diccionario. A diferencia del objeto de lista, el diccionario de Python no tiene un método *sort* incorporado. Necesitarás usar bucles FOR para ordenar los diccionarios ya sea por clave o por valor.

El diccionario a continuación es un resumen de la frecuencia de palabras de la canción *Shape of You* de Ed Sheeran. Cada clave es una palabra en la letra y el valor es el número de veces que esa palabra aparece en la letra.

```
word_freq = {'love': 25, 'conversation': 1, 'every': 6, "we're": 1,
'plate': 1, 'sour': 1, 'jukebox': 1, 'now': 11, 'taxi': 1, 'fast': 1,
'bag': 1, 'man': 1, 'push': 3, 'baby': 14, 'going': 1, 'you': 16,
"don't": 2, 'one': 1, 'mind': 2, 'backseat': 1, 'friends': 1, 'then':
3, 'know': 2, 'take': 1, 'play': 1, 'okay': 1, 'so': 2, 'begin': 1,
'start': 2, 'over': 1, 'body': 17, 'boy': 2, 'just': 1, 'we': 7,
'are': 1, 'girl': 2, 'tell': 1, 'singing': 2, 'drinking': 1, 'put': 3,
'our': 1, 'where': 1, "i'll": 1, 'all': 1, "isn't": 1, 'make': 1,
'lover': 1, 'get': 1, 'radio': 1, 'give': 1, "i'm": 23, 'like': 10,
'can': 1, 'doing': 2, 'with': 22, 'club': 1, 'come': 37, 'it': 1,
'somebody': 2, 'handmade': 2, 'out': 1, 'new': 6, 'room': 3, 'chance':
1, 'follow': 6, 'in': 27, 'may': 2, 'brand': 6, 'that': 2, 'magnet':
3, 'up': 3, 'first': 1, 'and': 23, 'pull': 3, 'of': 6, 'table': 1,
'much': 2, 'last': 3, 'i': 6, 'thrifty': 1, 'grab': 2, 'was': 2,
'driver': 1, 'slow': 1, 'dance': 1, 'the': 18, 'say': 2, 'trust': 1,
'family': 1, 'week': 1, 'date': 1, 'me': 10, 'do': 3, 'waist': 2,
'smell': 3, 'day': 6, 'although': 3, 'your': 21, 'leave': 1, 'want':
2, "let's": 2, 'lead': 6, 'at': 1, 'hand': 1, 'how': 1, 'talk': 4,
'not': 2, 'eat': 1, 'falling': 3, 'about': 1, 'story': 1, 'sweet': 1,
'best': 1, 'crazy': 2, 'let': 1, 'too': 5, 'van': 1, 'shots': 1, 'go':
2, 'to': 2, 'a': 8, 'my': 33, 'is': 5, 'place': 1, 'find': 1, 'shape':
6, 'on': 40, 'kiss': 1, 'were': 3, 'night': 3, 'heart': 3, 'for': 3,
'discovering': 6, 'something': 6, 'be': 16, 'bedsheets': 3, 'fill': 2,
'hours': 2, 'stop': 1, 'bar': 1}
```

Ordena las claves de `word_freq` de forma ascendente.

Por favor, crea un nuevo diccionario llamado `word_freq2` basado en `word_freq` con las claves ordenadas de forma ascendente.

Hay varias formas de lograr ese objetivo, pero muchas de ellas van más allá de lo que hemos cubierto hasta ahora en el curso. Hay una forma que describiremos empleando lo que has aprendido. Por favor, siéntete libre de usar esta forma o cualquier otra que desees.

1. Primero extrae las claves de `word_freq` y conviértelas en una lista llamada `keys`.
2. Ordena la lista `keys`.

3. Crea un diccionario vacío `word_freq2`.

4. Usa un bucle FOR para iterar cada valor en `keys`. Para cada clave iterada, encuentra el valor correspondiente en `word_freq` e inserta el par clave-valor en `word_freq2`.

 Documentación para un bucle for

Imprime `word_freq2` para examinar sus claves y valores. Tu salida debería ser:

```
{'a': 8, 'about': 1, 'all': 1, 'although': 3, 'and': 23, 'are': 1,
'at': 1, 'baby': 14, 'backseat': 1, 'bag': 1, 'bar': 1, 'be': 16,
'bedsheets': 3, 'begin': 1, 'best': 1, 'body': 17, 'boy': 2, 'brand':
6, 'can': 1, 'chance': 1, 'club': 1, 'come': 37, 'conversation': 1,
'crazy': 2, 'dance': 1, 'date': 1, 'day': 6, 'discovering': 6, 'do':
3, 'doing': 2, "don't": 2, 'drinking': 1, 'driver': 1, 'eat': 1,
'every': 6, 'falling': 3, 'family': 1, 'fast': 1, 'fill': 2, 'find':
1, 'first': 1, 'follow': 6, 'for': 3, 'friends': 1, 'get': 1, 'girl':
2, 'give': 1, 'go': 2, 'going': 1, 'grab': 2, 'hand': 1, 'handmade':
2, 'heart': 3, 'hours': 2, 'how': 1, 'i': 6, "i'll": 1, "i'm": 23,
'in': 27, 'is': 5, "isn't": 1, 'it': 1, 'jukebox': 1, 'just': 1,
'kiss': 1, 'know': 2, 'last': 3, 'lead': 6, 'leave': 1, 'let': 1,
"let's": 2, 'like': 10, 'love': 25, 'lover': 1, 'magnet': 3, 'make':
1, 'man': 1, 'may': 2, 'me': 10, 'mind': 2, 'much': 2, 'my': 33,
'new': 6, 'night': 3, 'not': 2, 'now': 11, 'of': 6, 'okay': 1, 'on':
40, 'one': 1, 'our': 1, 'out': 1, 'over': 1, 'place': 1, 'plate': 1,
'play': 1, 'pull': 3, 'push': 3, 'put': 3, 'radio': 1, 'room': 3,
'say': 2, 'shape': 6, 'shots': 1, 'singing': 2, 'slow': 1, 'smell': 3,
'so': 2, 'somebody': 2, 'something': 6, 'sour': 1, 'start': 2, 'stop':
1, 'story': 1, 'sweet': 1, 'table': 1, 'take': 1, 'talk': 4, 'taxi':
1, 'tell': 1, 'that': 2, 'the': 18, 'then': 3, 'thrifty': 1, 'to': 2,
'too': 5, 'trust': 1, 'up': 3, 'van': 1, 'waist': 2, 'want': 2, 'was':
2, 'we': 7, "we're": 1, 'week': 1, 'were': 3, 'where': 1, 'with': 22,
'you': 16, 'your': 21}

solution1={'a': 8, 'about': 1, 'all': 1, 'although': 3, 'and': 23,
'are': 1, 'at': 1, 'baby': 14, 'backseat': 1, 'bag': 1, 'bar': 1,
'be': 16, 'bedsheets': 3, 'begin': 1, 'best': 1, 'body': 17, 'boy': 2,
'brand': 6, 'can': 1, 'chance': 1, 'club': 1, 'come': 37,
'conversation': 1, 'crazy': 2, 'dance': 1, 'date': 1, 'day': 6,
'discovering': 6, 'do': 3, 'doing': 2, "don't": 2, 'drinking': 1,
'driver': 1, 'eat': 1, 'every': 6, 'falling': 3, 'family': 1, 'fast':
1, 'fill': 2, 'find': 1, 'first': 1, 'follow': 6, 'for': 3, 'friends':
1, 'get': 1, 'girl': 2, 'give': 1, 'go': 2, 'going': 1, 'grab': 2,
'hand': 1, 'handmade': 2, 'heart': 3, 'hours': 2, 'how': 1, 'i': 6,
"i'll": 1, "i'm": 23, 'in': 27, 'is': 5, "isn't": 1, 'it': 1,
'jukebox': 1, 'just': 1, 'kiss': 1, 'know': 2, 'last': 3, 'lead': 6,
'leave': 1, 'let': 1, "let's": 2, 'like': 10, 'love': 25, 'lover': 1,
'magnet': 3, 'make': 1, 'man': 1, 'may': 2, 'me': 10, 'mind': 2,
'much': 2, 'my': 33, 'new': 6, 'night': 3, 'not': 2, 'now': 11, 'of':
6, 'okay': 1, 'on': 40, 'one': 1, 'our': 1, 'out': 1, 'over': 1,
```

```
'place': 1, 'plate': 1, 'play': 1, 'pull': 3, 'push': 3, 'put': 3,
'radio': 1, 'room': 3, 'say': 2, 'shape': 6, 'shots': 1, 'singing': 2,
'slow': 1, 'smell': 3, 'so': 2, 'somebody': 2, 'something': 6, 'sour':
1, 'start': 2, 'stop': 1, 'story': 1, 'sweet': 1, 'table': 1, 'take':
1, 'talk': 4, 'taxi': 1, 'tell': 1, 'that': 2, 'the': 18, 'then': 3,
'thrifty': 1, 'to': 2, 'too': 5, 'trust': 1, 'up': 3, 'van': 1,
'waist': 2, 'want': 2, 'was': 2, 'we': 7, "we're": 1, 'week': 1,
'were': 3, 'where': 1, 'with': 22, 'you': 16, 'your': 21}

keys=list(word_freq.keys());

ord_keys=(keys.copy())
ord_keys.sort()

word_freq2=dict()
for i in ord_keys:
    word_freq2[i]=word_freq[i]

print(word_freq2==solution1)

True

print(word_freq2 is solution1)

False
```

Ordena los valores de `word_freq` de forma ascendente.

Ordenar los valores de un diccionario es más complicado que ordenar las claves porque los valores de un diccionario no son únicos. Por lo tanto, no puedes usar la misma forma en que ordenaste las claves del diccionario para ordenar los valores del diccionario.

La forma de ordenar un diccionario por valor es utilizar las funciones `sorted` y `operator.itemgetter`. El siguiente fragmento de código se te proporciona para que lo pruebes. Te dará una lista de tuplas en la que cada tupla contiene la clave y el valor de un elemento del diccionario. Y la lista está ordenada basada en el valor del diccionario (referencia).

```
import operator
sorted_tups = sorted(word_freq.items(), key=operator.itemgetter(1))
print(sorted_tups)
```

Por lo tanto, los pasos para ordenar `word_freq` por valor son:

- Utilizando `sorted` y `operator.itemgetter`, obtén una lista de tuplas de los pares clave-valor del diccionario que está ordenada por el valor.
- Crea un diccionario vacío llamado `word_freq2`.
- Itera la lista de tuplas. Inserta cada par clave-valor en `word_freq2` como un objeto.

Imprime `word_freq2` para confirmar que tu diccionario tiene sus valores ordenados. Tu salida debería ser:

```
{'conversation': 1, "we're": 1, 'plate': 1, 'sour': 1, 'jukebox': 1,
'taxi': 1, 'fast': 1, 'bag': 1, 'man': 1, 'going': 1, 'one': 1,
'backseat': 1, 'friends': 1, 'take': 1, 'play': 1, 'okay': 1, 'begin':
1, 'over': 1, 'just': 1, 'are': 1, 'tell': 1, 'drinking': 1, 'our': 1,
'where': 1, "i'll": 1, 'all': 1, "isn't": 1, 'make': 1, 'lover': 1,
'get': 1, 'radio': 1, 'give': 1, 'can': 1, 'club': 1, 'it': 1, 'out':
1, 'chance': 1, 'first': 1, 'table': 1, 'thrifty': 1, 'driver': 1,
'slow': 1, 'dance': 1, 'trust': 1, 'family': 1, 'week': 1, 'date': 1,
'leave': 1, 'at': 1, 'hand': 1, 'how': 1, 'eat': 1, 'about': 1,
'story': 1, 'sweet': 1, 'best': 1, 'let': 1, 'van': 1, 'shots': 1,
'place': 1, 'find': 1, 'kiss': 1, 'stop': 1, 'bar': 1, "don't": 2,
'mind': 2, 'know': 2, 'so': 2, 'start': 2, 'boy': 2, 'girl': 2,
'singing': 2, 'doing': 2, 'somebody': 2, 'handmade': 2, 'may': 2,
'that': 2, 'much': 2, 'grab': 2, 'was': 2, 'say': 2, 'waist': 2,
'want': 2, "let's": 2, 'not': 2, 'crazy': 2, 'go': 2, 'to': 2, 'fill':
2, 'hours': 2, 'push': 3, 'then': 3, 'put': 3, 'room': 3, 'magnet': 3,
'up': 3, 'pull': 3, 'last': 3, 'do': 3, 'smell': 3, 'although': 3,
'falling': 3, 'were': 3, 'night': 3, 'heart': 3, 'for': 3,
'bedsheets': 3, 'talk': 4, 'too': 5, 'is': 5, 'every': 6, 'new': 6,
'follow': 6, 'brand': 6, 'of': 6, 'i': 6, 'day': 6, 'lead': 6,
'shape': 6, 'discovering': 6, 'something': 6, 'we': 7, 'a': 8, 'like':
10, 'me': 10, 'now': 11, 'baby': 14, 'you': 16, 'be': 16, 'body': 17,
'the': 18, 'your': 21, 'with': 22, "i'm": 23, 'and': 23, 'love': 25,
'in': 27, 'my': 33, 'come': 37, 'on': 40}

solution2={'conversation': 1, "we're": 1, 'plate': 1, 'sour': 1,
'jukebox': 1, 'taxi': 1, 'fast': 1, 'bag': 1, 'man': 1, 'going': 1,
'one': 1, 'backseat': 1, 'friends': 1, 'take': 1, 'play': 1, 'okay':
1, 'begin': 1, 'over': 1, 'just': 1, 'are': 1, 'tell': 1, 'drinking':
1, 'our': 1, 'where': 1, "i'll": 1, 'all': 1, "isn't": 1, 'make': 1,
'lover': 1, 'get': 1, 'radio': 1, 'give': 1, 'can': 1, 'club': 1,
'it': 1, 'out': 1, 'chance': 1, 'first': 1, 'table': 1, 'thrifty': 1,
'driver': 1, 'slow': 1, 'dance': 1, 'trust': 1, 'family': 1, 'week':
1, 'date': 1, 'leave': 1, 'at': 1, 'hand': 1, 'how': 1, 'eat': 1,
'about': 1, 'story': 1, 'sweet': 1, 'best': 1, 'let': 1, 'van': 1,
'shots': 1, 'place': 1, 'find': 1, 'kiss': 1, 'stop': 1, 'bar': 1,
"don't": 2, 'mind': 2, 'know': 2, 'so': 2, 'start': 2, 'boy': 2,
'girl': 2, 'singing': 2, 'doing': 2, 'somebody': 2, 'handmade': 2,
'may': 2, 'that': 2, 'much': 2, 'grab': 2, 'was': 2, 'say': 2,
'waist': 2, 'want': 2, "let's": 2, 'not': 2, 'crazy': 2, 'go': 2,
'to': 2, 'fill': 2, 'hours': 2, 'push': 3, 'then': 3, 'put': 3,
'room': 3, 'magnet': 3, 'up': 3, 'pull': 3, 'last': 3, 'do': 3,
'smell': 3, 'although': 3, 'falling': 3, 'were': 3, 'night': 3,
'heart': 3, 'for': 3, 'bedsheets': 3, 'talk': 4, 'too': 5, 'is': 5,
'every': 6, 'new': 6, 'follow': 6, 'brand': 6, 'of': 6, 'i': 6, 'day':
6, 'lead': 6, 'shape': 6, 'discovering': 6, 'something': 6, 'we': 7,
'a': 8, 'like': 10, 'me': 10, 'now': 11, 'baby': 14, 'you': 16, 'be':
```

```
16, 'body': 17, 'the': 18, 'your': 21, 'with': 22, "i'm": 23, 'and':
23, 'love': 25, 'in': 27, 'my': 33, 'come': 37, 'on': 40}

import operator

sorted_tups = sorted(word_freq.items(), key=operator.itemgetter(1))
print(sorted_tups)

[('conversation', 1), ("we're", 1), ('plate', 1), ('sour', 1),
('jukebox', 1), ('taxi', 1), ('fast', 1), ('bag', 1), ('man', 1),
('going', 1), ('one', 1), ('backseat', 1), ('friends', 1), ('take',
1), ('play', 1), ('okay', 1), ('begin', 1), ('over', 1), ('just', 1),
('are', 1), ('tell', 1), ('drinking', 1), ('our', 1), ('where', 1),
("i'll", 1), ('all', 1), ("isn't", 1), ('make', 1), ('lover', 1),
('get', 1), ('radio', 1), ('give', 1), ('can', 1), ('club', 1), ('it',
1), ('out', 1), ('chance', 1), ('first', 1), ('table', 1), ('thrifty',
1), ('driver', 1), ('slow', 1), ('dance', 1), ('trust', 1), ('family',
1), ('week', 1), ('date', 1), ('leave', 1), ('at', 1), ('hand', 1),
('how', 1), ('eat', 1), ('about', 1), ('story', 1), ('sweet', 1),
('best', 1), ('let', 1), ('van', 1), ('shots', 1), ('place', 1),
('find', 1), ('kiss', 1), ('stop', 1), ('bar', 1), ("don't", 2),
('mind', 2), ('know', 2), ('so', 2), ('start', 2), ('boy', 2),
('girl', 2), ('singing', 2), ('doing', 2), ('somebody', 2),
('handmade', 2), ('may', 2), ('that', 2), ('much', 2), ('grab', 2),
('was', 2), ('say', 2), ('waist', 2), ('want', 2), ("let's", 2),
('not', 2), ('crazy', 2), ('go', 2), ('to', 2), ('fill', 2), ('hours',
2), ('push', 3), ('then', 3), ('put', 3), ('room', 3), ('magnet', 3),
('up', 3), ('pull', 3), ('last', 3), ('do', 3), ('smell', 3),
('although', 3), ('falling', 3), ('were', 3), ('night', 3), ('heart',
3), ('for', 3), ('bedsheets', 3), ('talk', 4), ('too', 5), ('is', 5),
('every', 6), ('new', 6), ('follow', 6), ('brand', 6), ('of', 6),
('i', 6), ('day', 6), ('lead', 6), ('shape', 6), ('discovering', 6),
('something', 6), ('we', 7), ('a', 8), ('like', 10), ('me', 10),
('now', 11), ('baby', 14), ('you', 16), ('be', 16), ('body', 17),
('the', 18), ('your', 21), ('with', 22), ("i'm", 23), ('and', 23),
('love', 25), ('in', 27), ('my', 33), ('come', 37), ('on', 40)]

word_freq2=dict()
word_freq2

{}

word_freq2.update(sorted_tups)

print(word_freq2 == solution2)

True

print(word_freq2 is solution2)

False
```