

Python / Guía 5

Colecciones

de datos II

## 

## 

## OBJETIVOS DE LA GUÍA

En esta guía aprenderemos a:

* Manejar los diccionarios
* Iterar sobre las colecciones de datos
* Convertir los tipos de datos
* Crear comprensiones de listas

# 

# Diccionarios

## ¿Qué es un diccionario y cómo se crea?

Los diccionarios son una colección de datos indexados por claves que tienen asociados valores.

Los diccionarios en Python tienen las siguientes características:

* Mantienen el orden, según se van creando las claves.
* Son mutables: se puede añadir, borrar y modificar sus elementos.
* Las claves son únicas.
* Las claves tienen la restricción de que sus elementos deben ser "hashables", es decir, no pueden ser listas, conjuntos y diccionarios; pero sí pueden ser enteros, flotantes, cadenas, booleanos y tuplas.
* Tienen un acceso muy rápido a sus elementos, debido a la forma en la que están implementados internamente.

Su estructura lógica es:

<variable> = {<clave hashable>: <elemento>, <...>}

Prueba el siguiente ejemplo:

mi\_diccionario = {"sal": 10.50, "pan": 20.00, "leche": 25.25}  
 print(mi\_diccionario)  
 print(type(mi\_diccionario))

Y este otro:

mi\_diccionario\_2 = dict(sal=20, pan=40, leche=35.25)  
 print(mi\_diccionario\_2)  
 print(type(mi\_diccionario\_2))

## 

## ¿Cómo invoco sus valores?

Puedes invocar el diccionario y el índice: dict[<índice>], pero recuerda que la clave es el índice, y la clave es única y debe ser un elemento hashable, por lo que debes llamar el índice con el nombre de la clave:

**Ejecuta el siguiente código:**

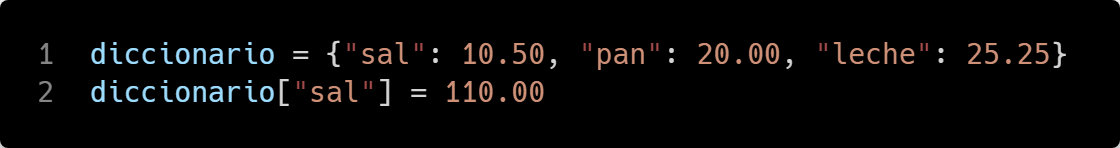
## 

## 

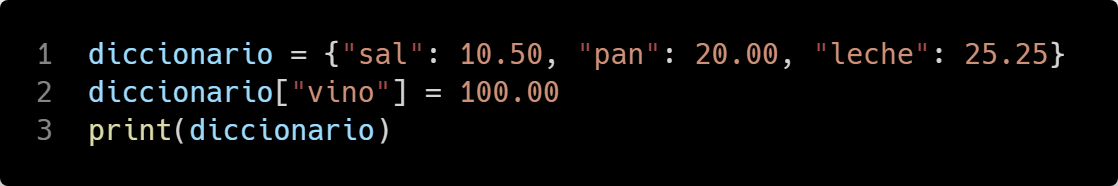
## ¿Cómo se modifica un elemento de un diccionario?

Invoca el diccionario y su clave, y asígnale un nuevo valor:

**Ejecuta el siguiente código:**



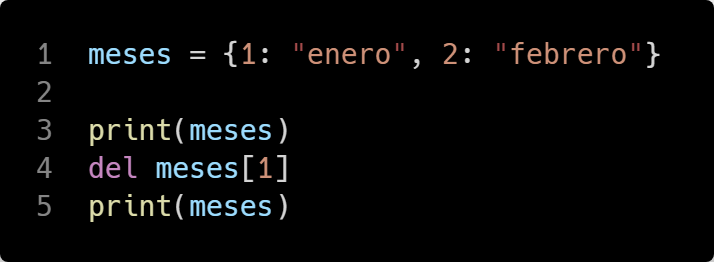
## ¿Cómo agrego un elemento a un diccionario?

**Ejecuta el siguiente código:**

## ¿Cómo elimino un elemento de un diccionario?

Usa la instrucción 'del' seguido del nombre del diccionario y su clave:

**Ejecuta el siguiente código:**

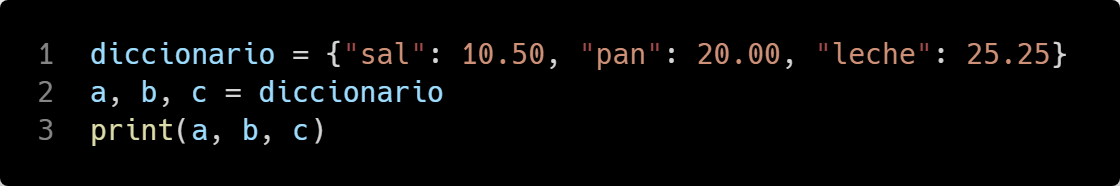


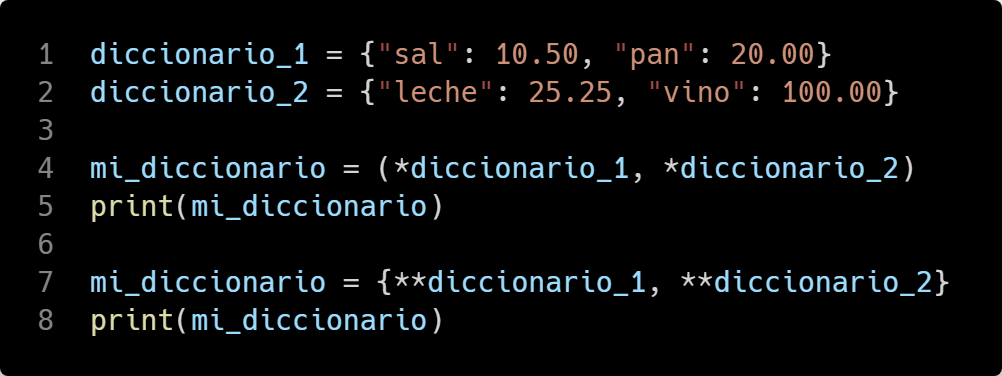
## ¿Cómo desempacar diccionarios?

**¡MANOS A LA OBRA!**

**Ejercicio 1**

Ejecuta el código y responde al título de este apartado, y por qué:



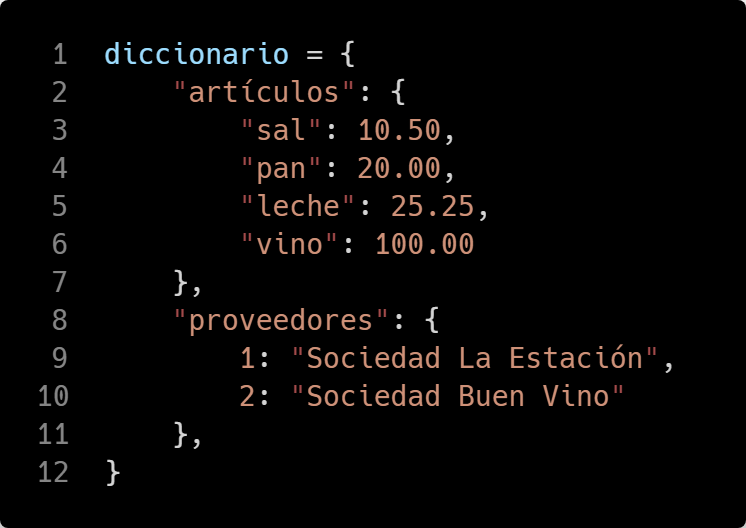


## 

## ¿Cómo anidar en diccionarios?

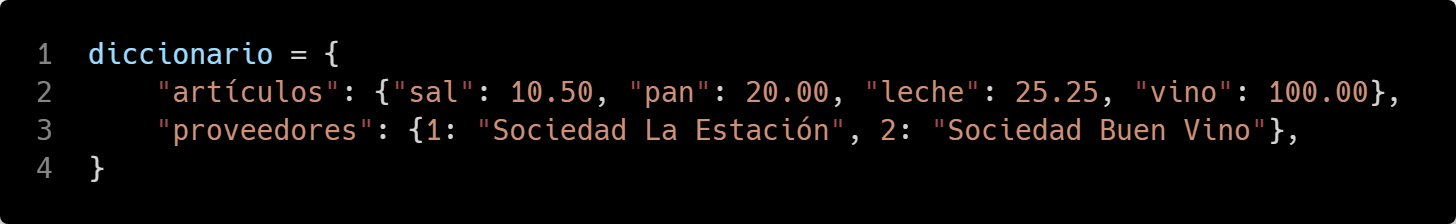
Los diccionarios pueden anidar más diccionarios. Puedes escribirlo de esta forma:

**Observa el siguiente código:**



O de esta forma:

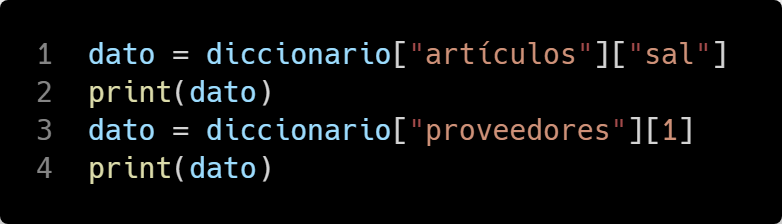
**Observa el siguiente código:**



O como mejor prefieras visualizar el diccionario...

Y puedes invocar las claves anidadas de la siguiente forma:

**Ejecuta el siguiente código (habiendo creado el diccionario anterior):**



## 

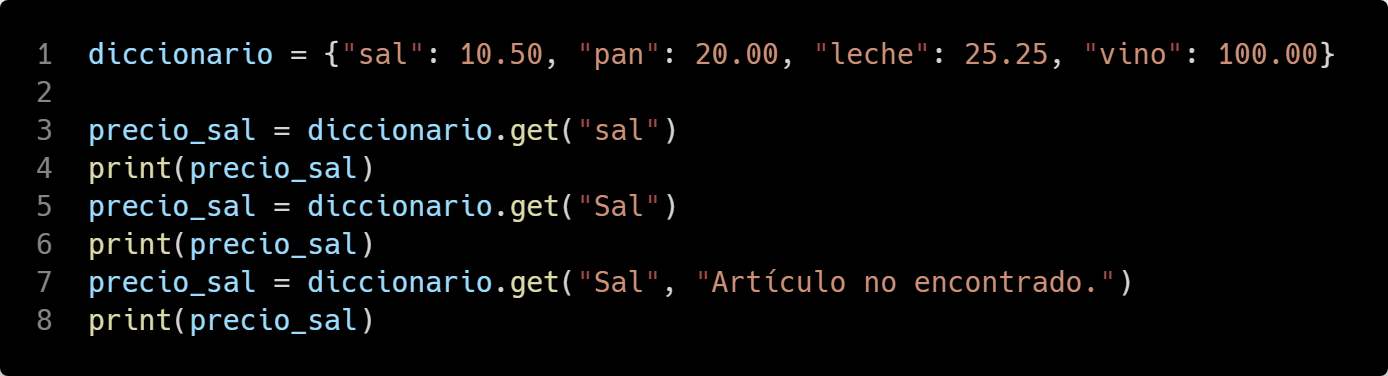
## Funciones de diccionarios

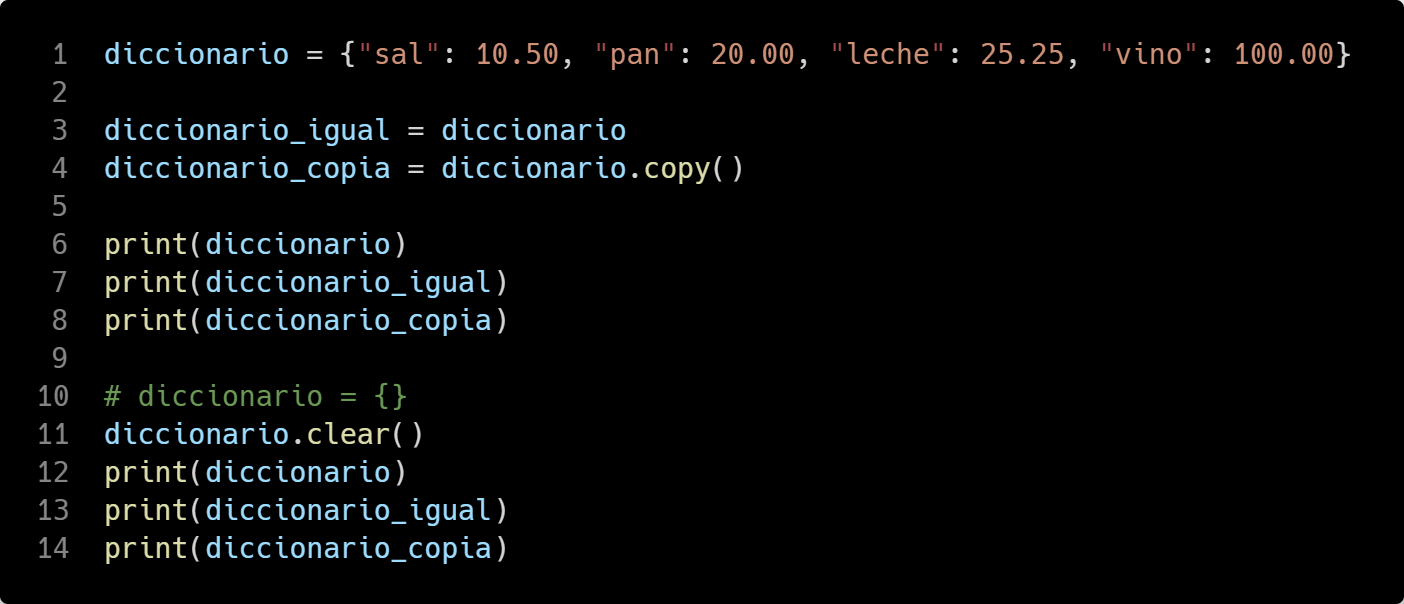
* get
* copy
* clear
* fromkeys
* keys
* values
* items
* update
* pop
* popitem
* setdefault

**¡MANOS A LA OBRA!**

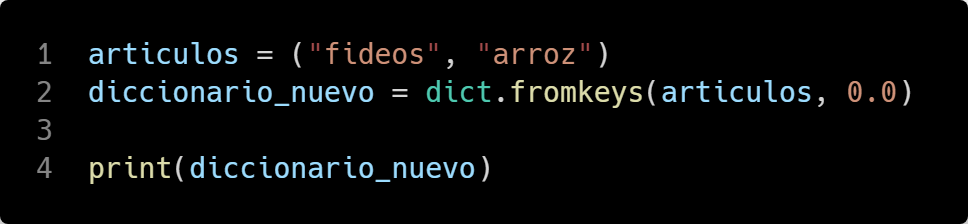
**Ejercicio 2**

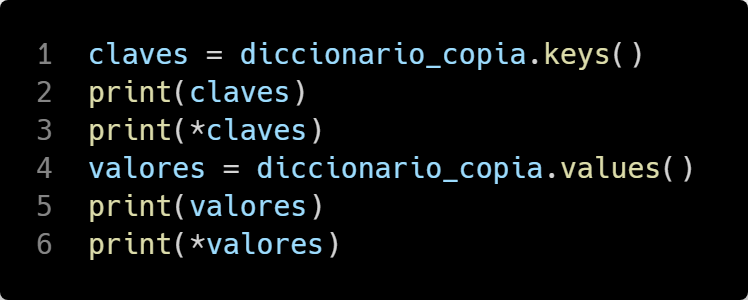
Ejecuta las siguientes partes de código, de forma ordenada, e intenta explicar:

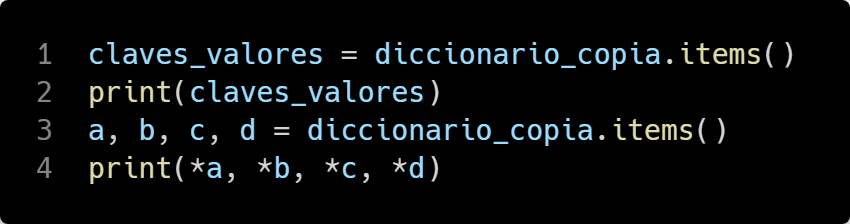




| Al igual que ocurre con las listas, si hacemos un cambio en un diccionario, se verá reflejado en todas las variables que hagan referencia al mismo. Esto se deriva de su propiedad de ser mutable. Viene a nuestra ayuda 'copy' y 'clear' |
| --- |



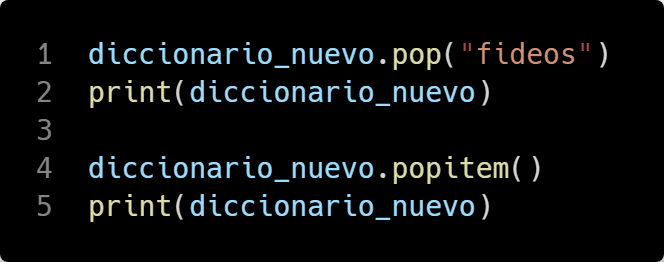




### 

### 

### 



### 

### 

### 

## Revisemos lo aprendido hasta aquí

* Crear diccionarios
* Invocar sus elementos
* Desempacar diccionarios
* Anidar en diccionarios
* Utilizar las funciones incorporadas de los diccionarios

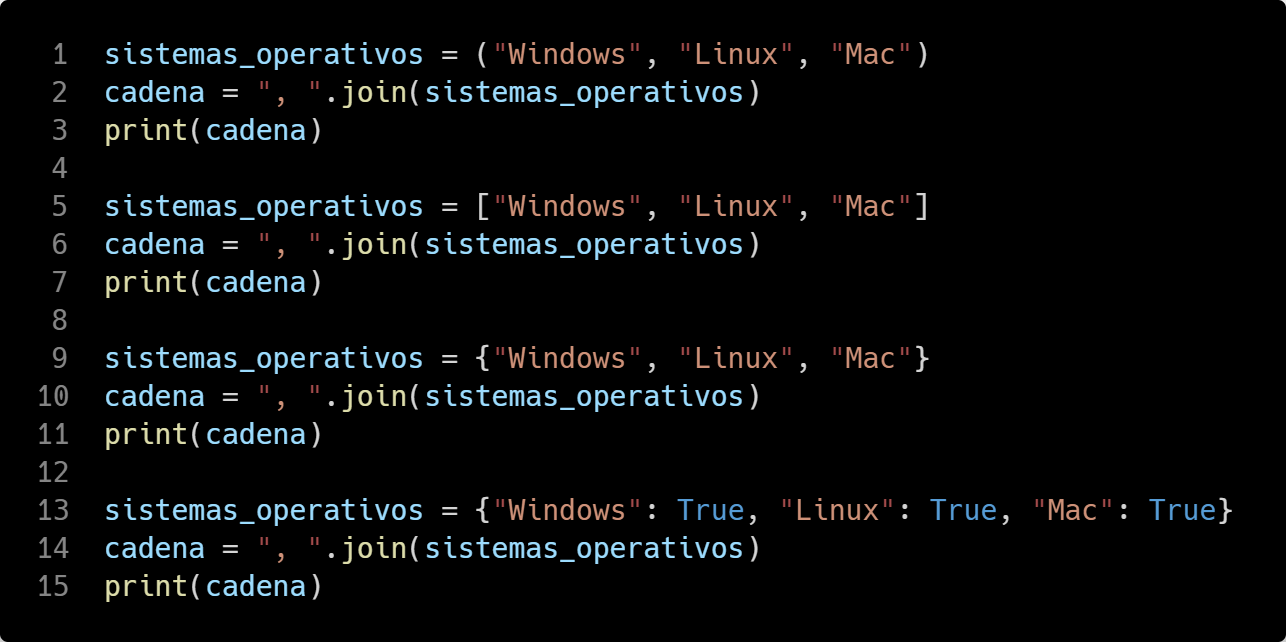
# 

# Funciones Join y Split

## Join

Join es una función propia de las **cadenas,** que trabaja con **colecciones de datos.** Su objetivo es formar una cadena de caracteres, con los elementos de la colección. En el caso de los diccionarios, tomará como elementos las claves de los mismos. Los elementos se guardarán en la cadena, separados por el carácter que definamos.

**Ejecuta el siguiente código**

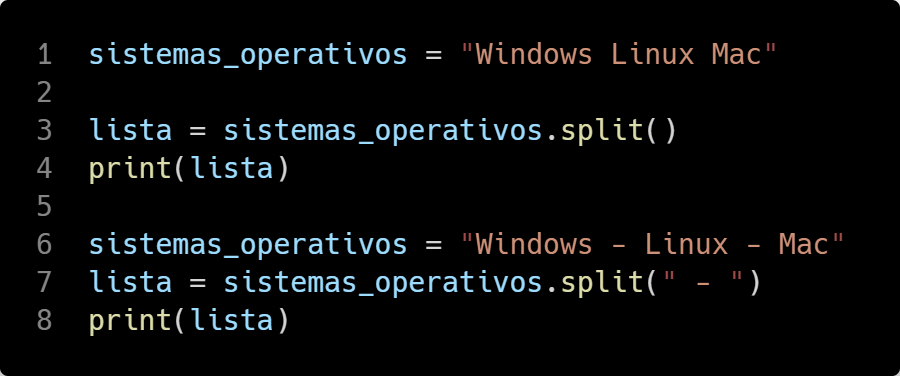


## 

## Split

Split es también una función propia de las **cadenas,** que trabaja con **listas.** Su objetivo es separar una cadena de caracteres, y colocar los elementos separados en una lista.

**Ejecuta el siguiente código**

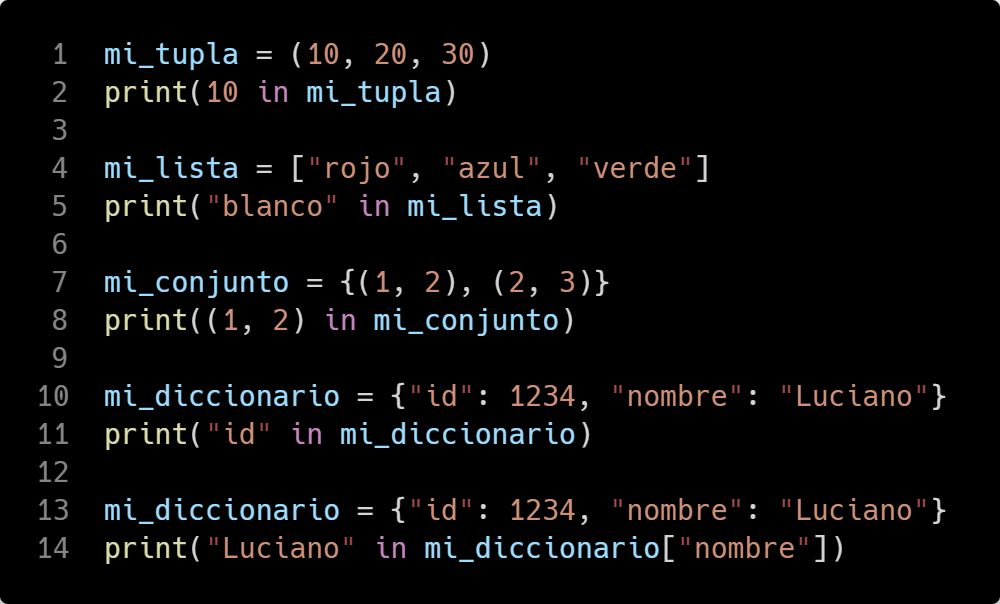


# 

# Iteración sobre colecciones de datos

## In

Con la instrucción 'in' podemos consultar a Python si un elemento existe en una colección, y se nos devolverá True o False.

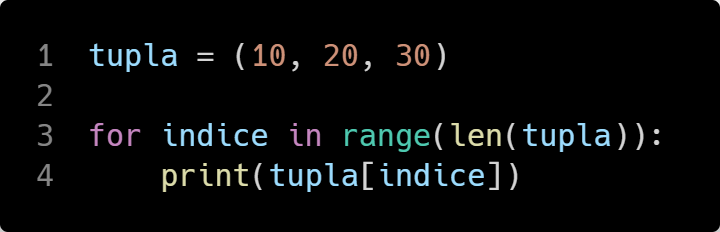
**Ejecuta el siguiente código:**

## 

## For

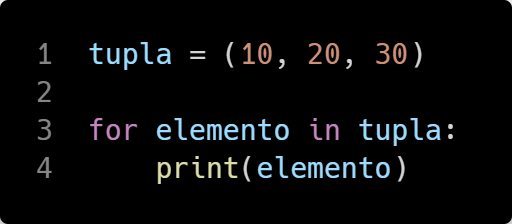
Con la instrucción **'for'** (el iterador) podemos recorrer una colección (toda colección es iterable). De manera que, podemos trabajar con cada elemento devuelto por for.

**Observa el siguiente código:**

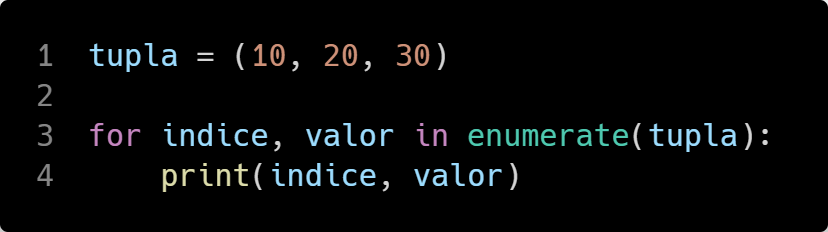


Puedes conseguir el mismo resultado con código más limpio:

**Ejecuta el siguiente código:**



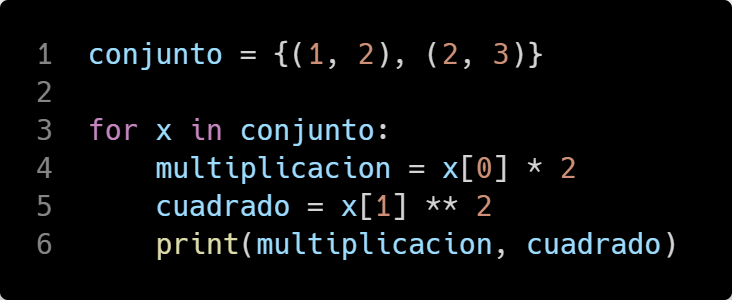
Si necesitas un índice, puedes utilizar la función **'enumerate':**



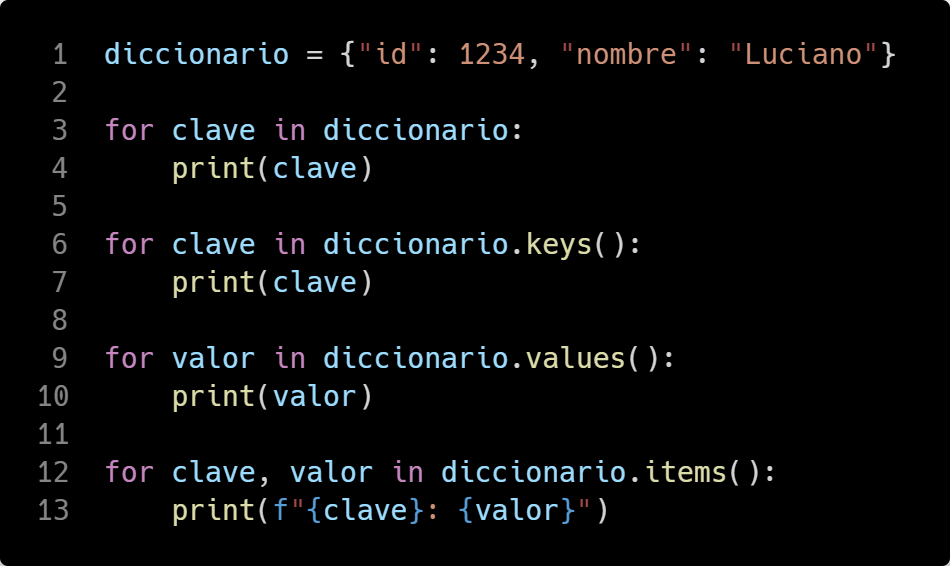
**Ejecuta el siguiente código:**



**Ejecuta el siguiente código:**



**Ejecuta el siguiente código:**



# Conversión de tipos en colecciones

Puedes convertir una colección a una **tupla** de la siguiente forma:

x = [1, 2, 3]  
 y = tuple(x)

x = {1, 2, 3}  
 y = tuple(x)

x = {1: "uno", 2: "dos", 3: "tres"}  
 y = tuple(x) # toma las claves del diccionario

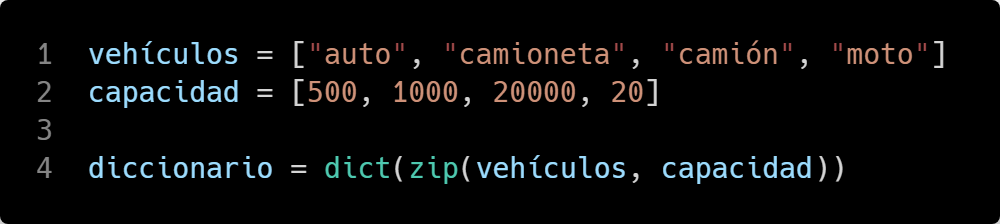
Puedes convertir una colección a una **lista,** se procede de modo semejante:

list(x)

Una colección a un conjunto:  
 set(x)

Una colección a un diccionario: Una de las formas más usadas es utilizar la función incorporada **'zip':**

**Observa el siguiente código:**



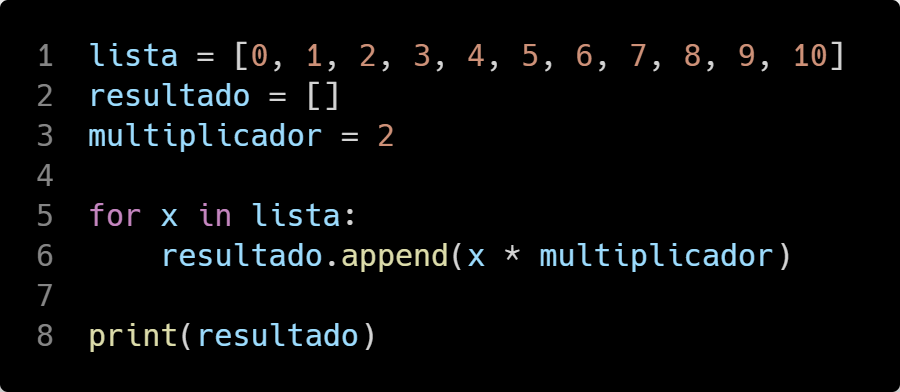
# 

# Comprensión de listas

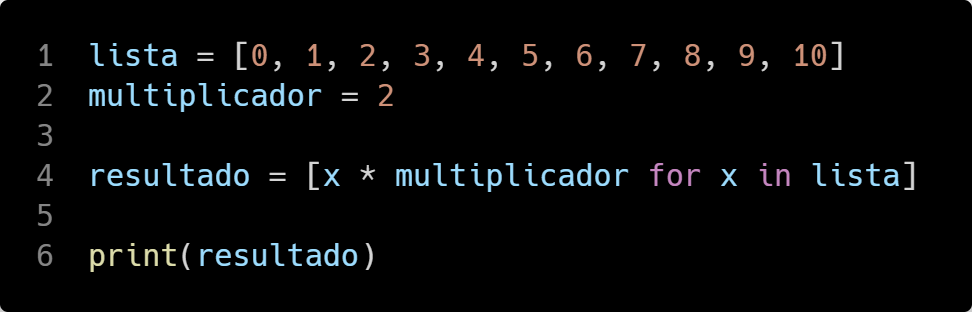
La comprensión de listas es una construcción de Python que nos permite crear listas a partir de otras listas, tuplas y cualquier iterable. Nos permite escribir de forma más concisa y clara. Su estructura es:

[<variable> for <variable> in <iterable> if <variable> <expresión>]

**Ejecuta el siguiente código:**

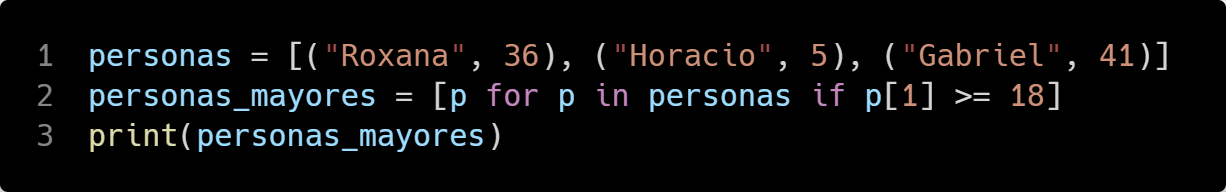


**Ejecuta el siguiente código:**



| Intenta [x for x in lista] o juega con las variables para comprender mejor cómo trabaja la comprensión de listas. |
| --- |

**Ejecuta el siguiente código:**



Revisemos lo aprendido hasta aquí

* Manejar cadenas con 'join' y 'split'
* Preguntar a Python si un elemento existe en una colección con 'in'
* Iterar sobre colecciones con 'for'
* Convertir tipos de datos entre las colecciones
* Utilizar las funciones 'zip' y 'enumerate'
* Crear comprensiones de listas