

Python / Guía 4

Colecciones

de datos I

## 

## 

## OBJETIVOS DE LA GUÍA

En esta guía aprenderemos a:

* Ambientarse en la escritura de código Python

en Visual Studio Code

* Crear colecciones de datos: tuplas, listas y conjuntos
* Invocar sus elementos, modificarlos, desempacarlos

y anidarlos

* Usar sus funciones incorporadas

## 

# 

## ¿Cuáles son las colecciones de datos en Python?

Las colecciones de datos sirven para guardar un conjunto de elementos (pueden guardar objetos, variables, datos de tipo int, float, complex, bool, str, y otros objetos. En Python, las colecciones que vienen incorporadas son:

* Las tuplas: tuple
* Las listas: list
* Los conjuntos: set
* Los diccionarios: dict

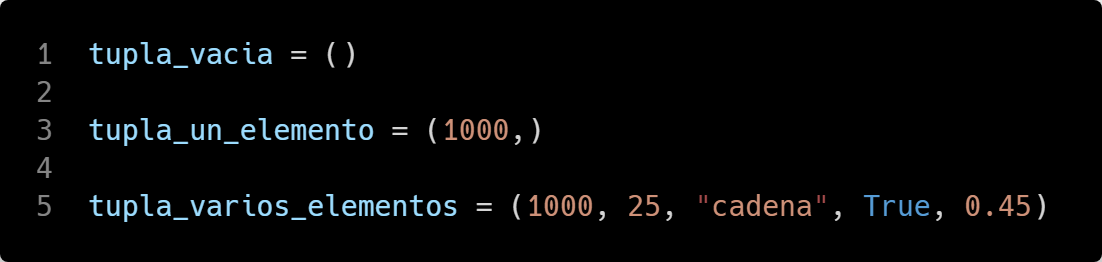
# Tuplas

## ¿Qué son las tuplas y cómo se crean?

Las tuplas son un tipo de colección de datos, significa que puede albergar diferentes tipos de datos como números, cadenas, booleanos y también tuplas y otros tipos de colecciones y objetos.

Una tupla se declara usando paréntesis, y los elementos se separan con una coma. Su estructura lógica es:

<variable> = (<elemento>, <elemento>, <...>).

**¿Necesitas un ejemplo?**

### 

| En la tupla con un elemento, no te olvides de escribir la coma, si no, Python va a creer que declaras un número, un texto, un booleano, etc.  Prueba ejecutar:  prueba = (1000)  print(type(prueba))  prueba = (1000,)  print(type(prueba)) |
| --- |

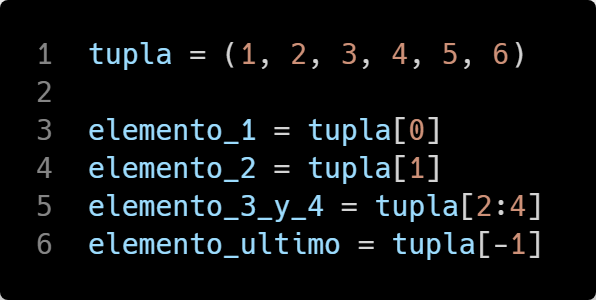
También se puede crear **tuplas sin paréntesis.** Prueba el siguiente código:

mi\_tupla = -.3, -.12, -.34  
 print(mi\_tupla)  
 print(type(mi\_tupla))

## 

## ¿Puedo usar el slicing?

Se puede obtener un elemento de una tupla o un conjunto de elementos utilizando, al igual que en las cadenas, el método slicing, que consiste en usar los corchetes después de la variable y escribir llamar al índice:

**¿Necesitas un ejemplo?**

### 

### 

### 

## ¿Cómo se modifica un elemento de una tupla?

No se pueden modificar. Las tuplas son estructuras de datos **inmutables,** ya que, una vez ingresados los datos, no pueden ser modificados. Ve qué ocurre si lo intentamos:

tupla = (1, 2, 2, 4, 5, 6)  
 tupla[2] = 3

Python lanzará un error TypeError.

**¡MANOS A LA OBRA!**

**Ejercicio 1**

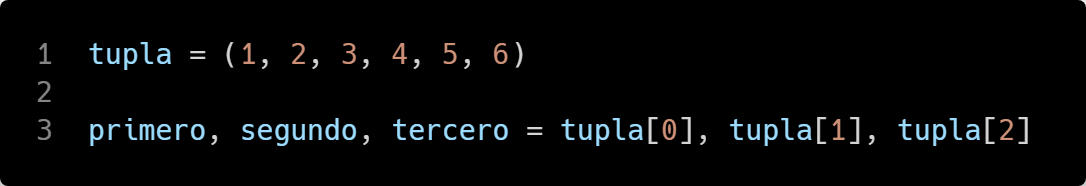
A partir de la siguiente tupla, crea una variable que contenga los dos primeros elementos de la tupla, y otra, que contenga los dos últimos elementos utilizando el método slicing.

primos = (2, 3, 5, 7, 11, 13, 17, 19, 23)

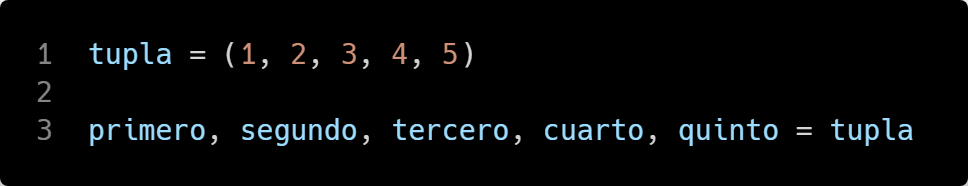
## 

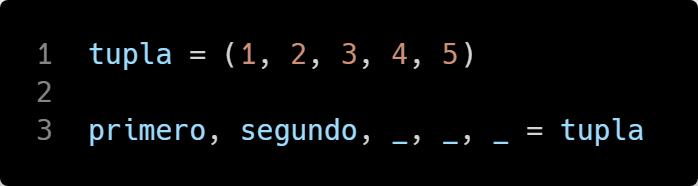
## ¿Existen otras formas de manejar los elementos?

Se pueden hacer varias asignaciones utilizando slicing en una línea:

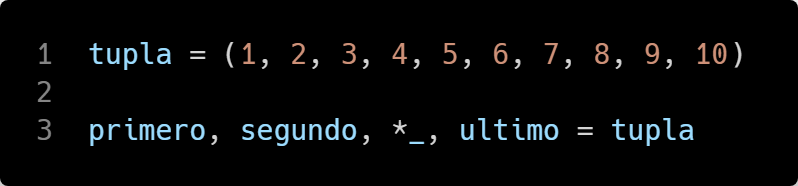


O simplemente no usar el slicing:



Si solo necesitas dos variables, puedes usar los guiones bajos, como lo siguiente:

Ahora, si tienes una tupla mucho más grande y necesitas obtener tres elementos, por ejemplo, el primero, el segundo y el último, puede usar el asterisco junto al guión bajo, como comodín de omisión \*\_ .



| Este procedimiento se llama desempacar una tupla. |
| --- |

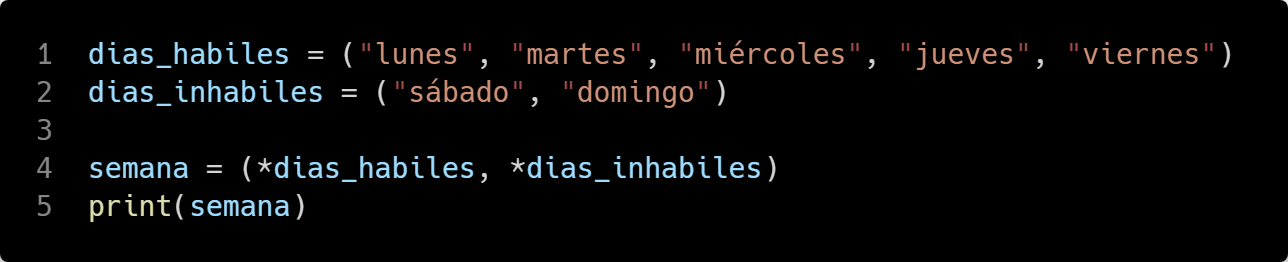
## 

## ¿Qué es desempacar?

Desempacar es tomar los elementos internos de una colección y "dejarlos sueltos". Se utiliza el asterisco \* para desempacar elementos de una colección. Prueba el siguiente código.

dias\_habiles = ("lunes", "martes", "miércoles", "jueves", "viernes")  
dias\_inhabiles = ("sábado", "domingo")  
semana = (dias\_habiles, dias\_inhabiles)  
print(semana)

Verás que la salida genera una única tupla que anida dos tuplas diferentes. Pero si queremos una sola tupla con todos los días de la semana, como elementos sueltos, necesitamos desempacar de la siguiente manera:



**¡MANOS A LA OBRA!**

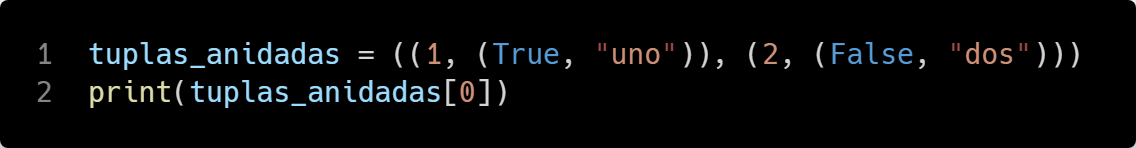
**Ejercicio 2**

A partir de la siguiente tupla, desempaca el último elemento sin usar slicing.

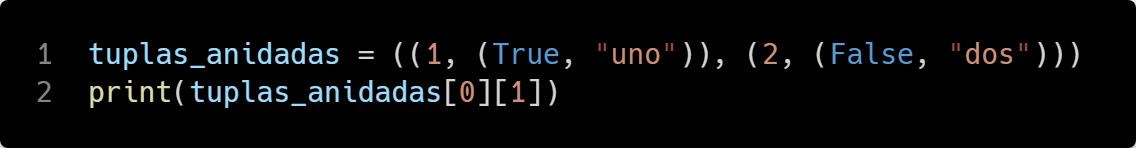
conocimientos = ("Python", "Javascript", "HTML", "CSS", "SQL")

## ¿Qué son las anidaciones?

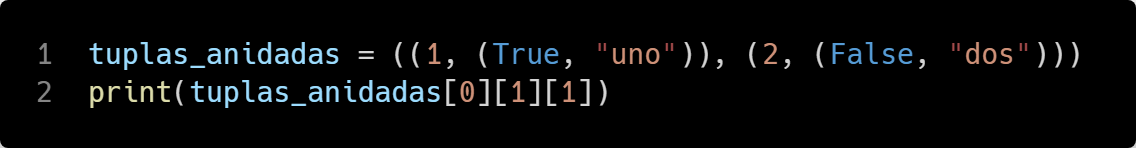
Una tupla puede anidar otras tuplas, listas, conjuntos y diccionarios, en varios niveles de profundidad, como ocurrió en el ejemplo de arriba. Para acceder a tuplas anidadas, se debe hacer de la siguiente manera:



La salida es: (1, (True, 'uno'))



La salida es: (True, 'uno')



La salida es: 'uno'

| Recuerda que puedes desempacar:  tupla 1, tupla 2 = tuplas anidadas |
| --- |

**¡MANOS A LA OBRA!**

**Ejercicio 3**

A partir de la siguiente tupla, guarda en una variable la tupla que contiene el booleano True mediante slicing:

validaciones = ((False, False), (False, True), (False, False))

## ¿Más funciones de tuplas?

### Operaciones de tuplas:

* Suma +
* Multiplicación \*
* Comparación < > == !=

### Funciones incorporadas en las tuplas:

* **count:** tupla.count(<elemento>)

Sirve para contar cuántas veces se encuentra un elemento en la tupla.

* **index:** tupla.index(<elemento>,<desde: opcional>)

Sirve para que Python nos devuelva en qué índice está el elemento.

**¡MANOS A LA OBRA!**

**Ejercicio 4**

Ejecuta el siguiente código e intenta explicarlo:

vocales\_1 = ("a", "e", "i")  
 vocales\_2 = ("o", "u")  
 print(vocales\_1 + vocales\_2)  
 print(vocales\_1 \* 3)

numeros\_1 = (1, 2, 2, 4, 5)  
 numeros\_2 = (1, 2, 3, 4, 5)  
gt6 print(numeros\_1 >= numeros\_2)

**Ejercicio 5**

Ejecuta el siguiente código e intenta explicarlo:

tupla\_texto = ("agua", "hidrógeno", "oxígeno", "agua")  
 print(tupla\_texto.count("agua"))  
 print(tupla\_texto.index("agua"))  
 print(tupla\_texto.index("agua", 1))

## Revisemos lo aprendido hasta aquí

* Comprender qué son las colecciones de datos
* Crear tuplas
* Invocar sus elementos
* Desempacar tuplas
* Anidar en tuplas
* Utilizar las funciones incorporadas de las tuplas

# Listas

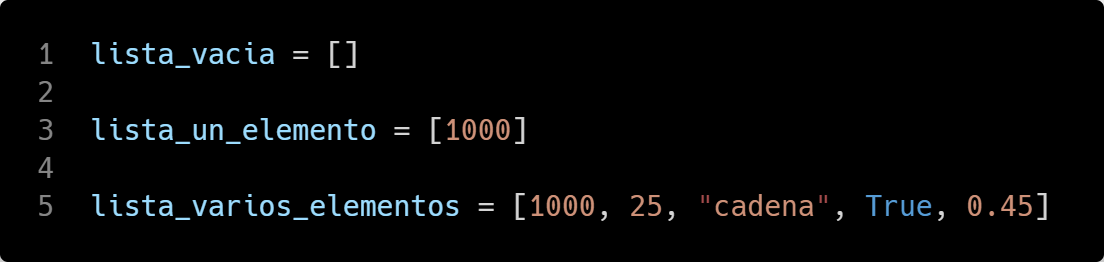
## ¿Qué son las listas y cómo se crean?

Las listas permiten almacenar objetos mediante un orden definido y con posibilidad de duplicados. Las listas son estructuras de datos mutables, lo que significa que podemos añadir, eliminar o modificar sus elementos. Su estructura lógica es:

<variable> = [<elemento>, <elemento>, <...>]

Prueba el siguiente ejemplo:

mi\_lista = ["hola", "lista"]  
 print(mi\_lista)  
 print(type(mi\_lista))

**¿Necesitas otro ejemplo?**

### 

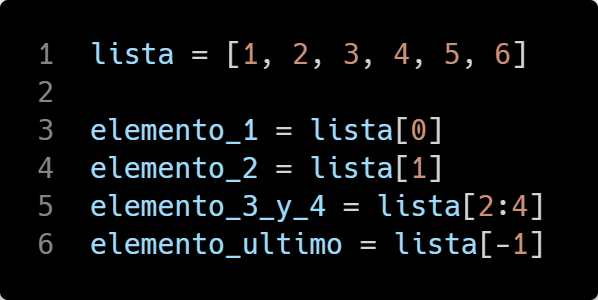
### 

| Tan fácil como cuando creas una tupla...Cuando tienes un solo elemento, no es necesario la coma al final del último elemento como sucede con la tupla. |
| --- |

## 

## ¿Puedo usar el slicing?

Al igual que las tuplas, puedes usar slicing:

Ejecuta el siguiente código:

### 

### 

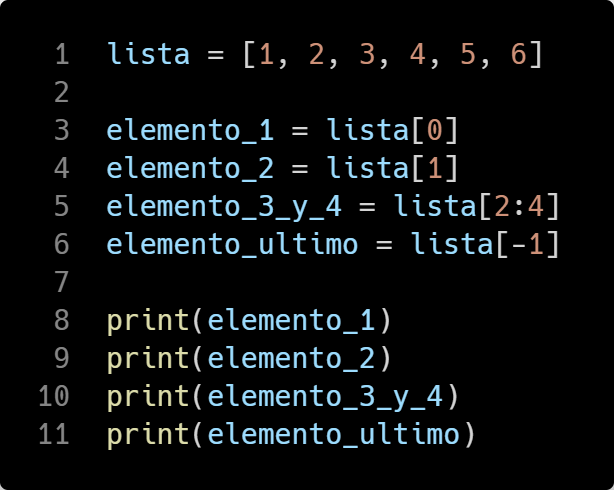
### 

## 

## ¿Cómo se modifica un elemento de una lista?

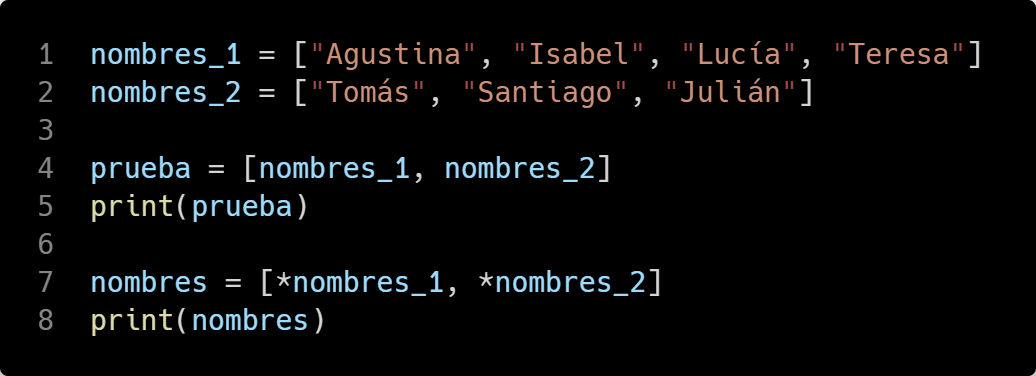
Para modificar un elemento de una lista debes invocar el índice del elemento a cambiar.

Ejecuta el siguiente código:



## ¿Cómo desempacar listas?

Al igual que las tuplas, puedes desempacar una lista:

Ejecuta el siguiente código:

### 

### 

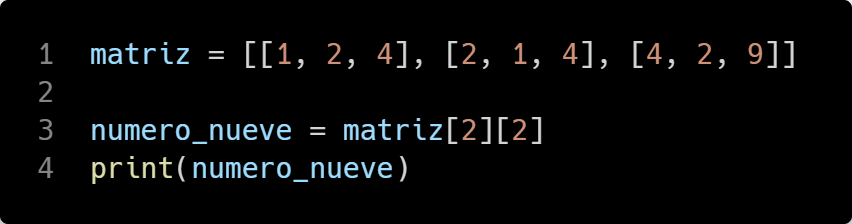
### 

### 

## 

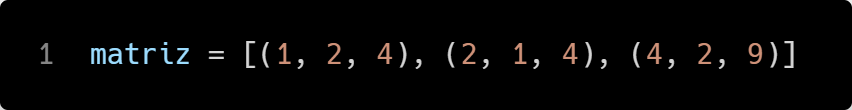
## ¿Cómo anidar en listas?

Al igual que las tuplas, puedes anidar e invocar los índices según la profundidad de la anidación.

Ejecuta el siguiente código:

### 

### 

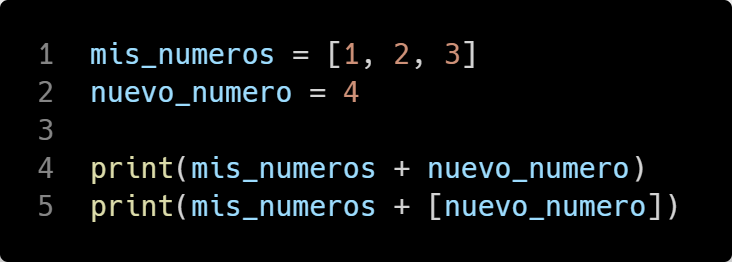
Observa el siguiente código:

| Puedes anidar tuplas. |
| --- |

## Funciones de listas

### Operaciones que podemos realizar, al igual que las tuplas:

* Suma +
* Multiplicación \*
* Comparación < > == !=

Repara el siguiente código:

### 

### Funciones incorporadas en las tuplas:

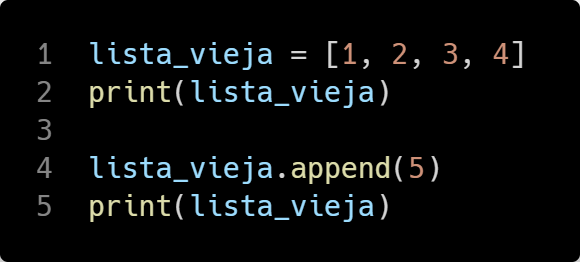
* append
* copy
* clear
* extend
* count
* index
* insert
* pop
* remove
* reverse
* sort

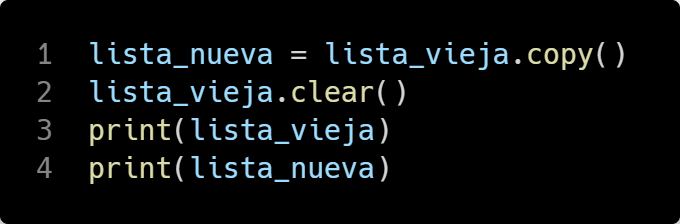
| Cada colección de datos tiene sus propias funciones incorporadas: no podemos usar las funciones de las listas en otro tipo de dato como las tuplas, aunque a veces sus nombres sean iguales, porque funcionan internamente de forma diferente. |
| --- |

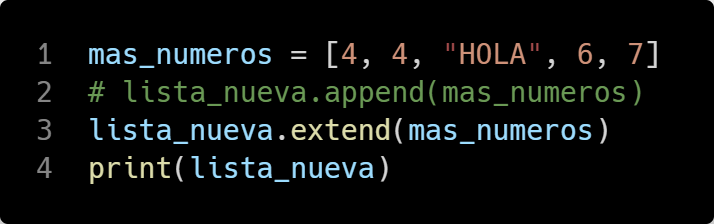
**¡MANOS A LA OBRA!**

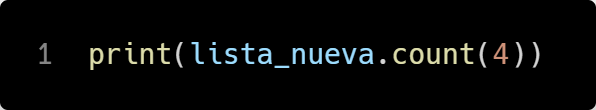
**Ejercicio 6**

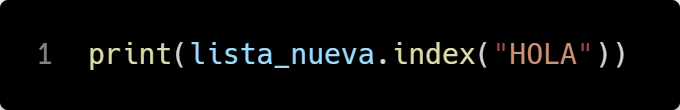
Ejecuta las siguientes partes de código, de forma ordenada, e intenta explicar que hace el código con tu equipo (prueba quitar el comentario de código donde exista, para ver qué sucede):

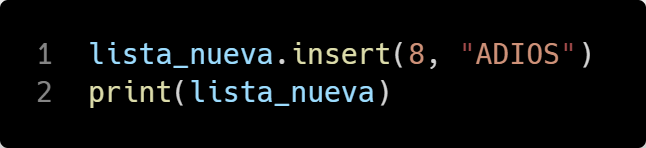


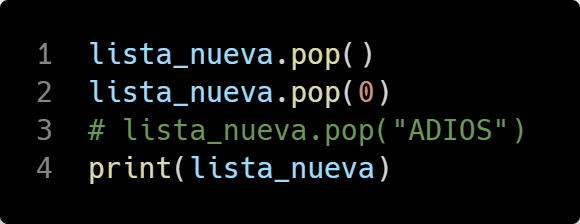


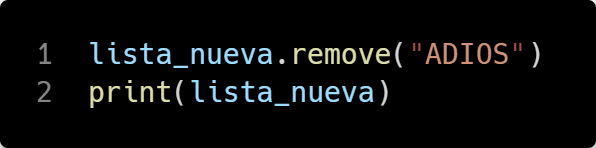


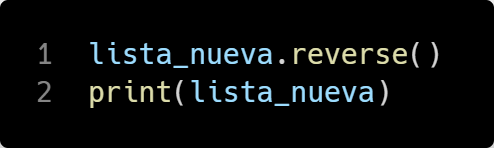














## Revisemos lo aprendido hasta aquí

* Crear listas
* Invocar sus elementos
* Desempacar listas
* Anida en listas
* Utilizar las funciones incorporadas de las listas

# Conjuntos

## ¿Qué son los conjuntos y cómo se crean?

Los conjuntos son una colección de datos de valores únicos y sin orden establecido. La única restricción que tiene es que sus elementos deben ser "hashables", es decir, no pueden contener listas, conjuntos y diccionarios; pero sí pueden contener enteros, flotantes, cadenas y tuplas.

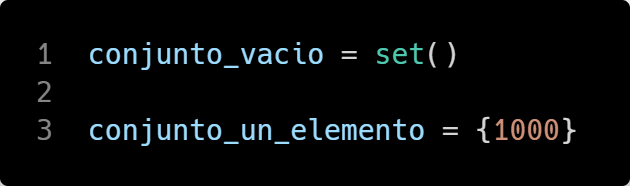
Su estructura lógica es:

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

Prueba el siguiente ejemplo:

mi\_conjunto = {10, 1.2, "hola", True, False}  
 print(mi\_conjunto)  
 print(type(mi\_conjunto))

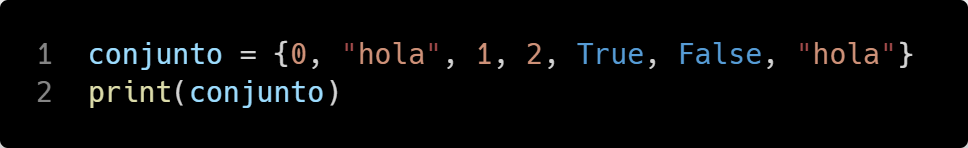
**¿Necesitas otro ejemplo?**



### 

**¡MANOS A LA OBRA!**

**Ejercicio 7**

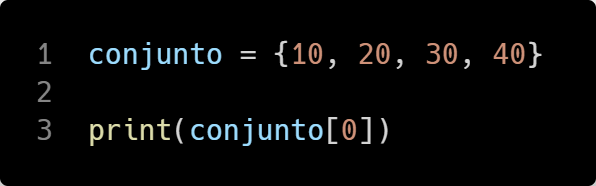
Ejecuta el siguiente código e intenta explicarlo:

### 

## ¿Puedo usar slicing?

**¡MANOS A LA OBRA!**

**Ejercicio 8**

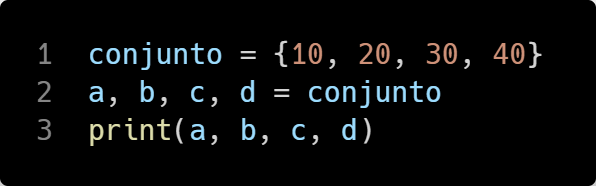
Ejecuta el código y responde al título de este apartado, y por qué:

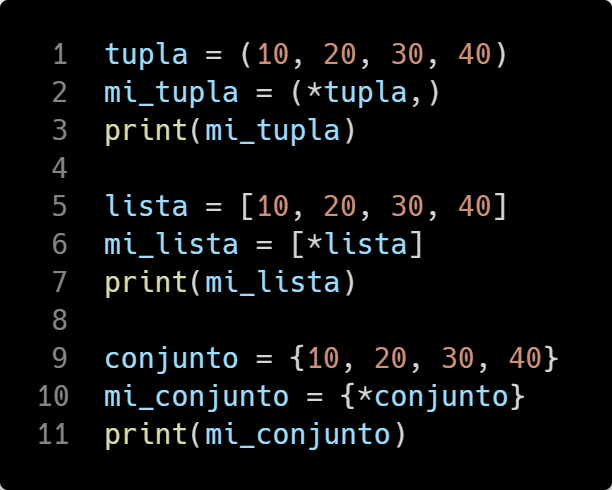
## ¿Cómo desempacar conjuntos?

**¡MANOS A LA OBRA!**

**Ejercicio 9**

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

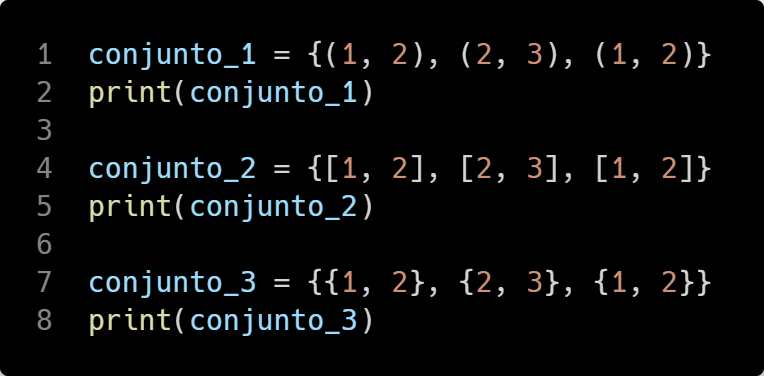




## ¿Cómo anidar en conjuntos?

**¡MANOS A LA OBRA!**

**Ejercicio 10**

Ejecuta el código y responde al título de este apartado, y por qué (no todo va a funcionar):

# 

# Funciones de conjuntos

No podemos usar las operaciones aritméticas en el mismo sentido que las tuplas y listas. Tenemos funciones especiales para los conjuntos.

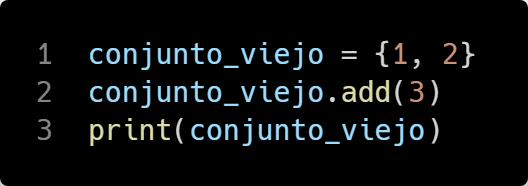
### Funciones incorporadas en los conjuntos:

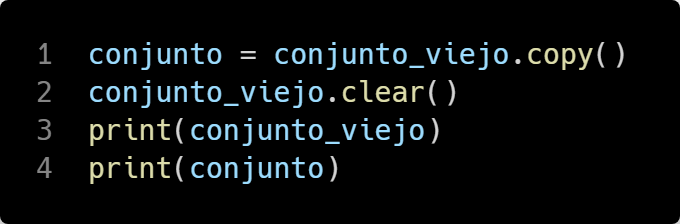
* add
* copy
* clear
* discard
* pop
* remove
* update

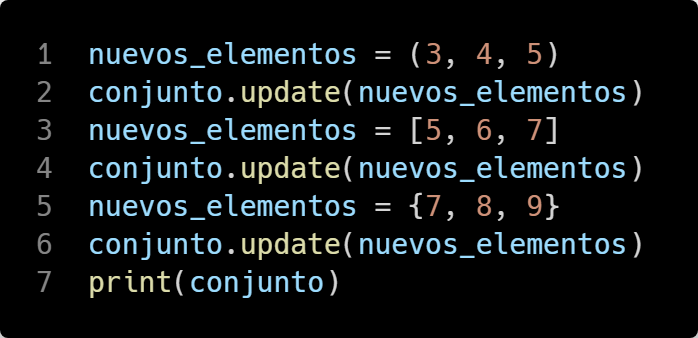
**¡MANOS A LA OBRA!**

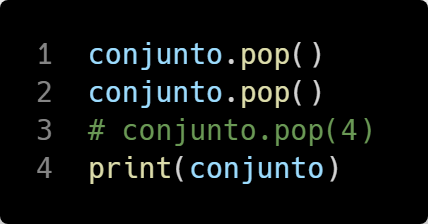
**Ejercicio 11**

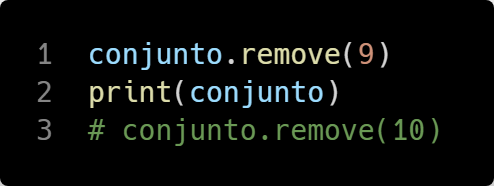
Ejecuta las siguientes partes de código, de forma ordenada, e intenta explicar (prueba quitar el comentario de código donde exista, para ver qué sucede):

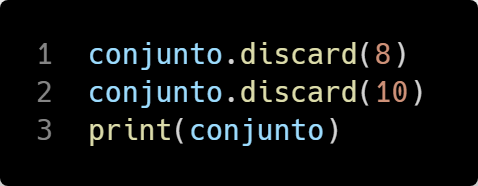


****

****

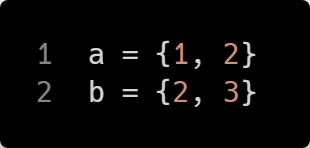
****

****

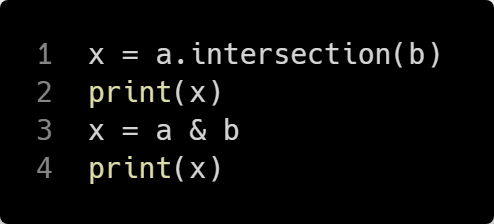
****

## Más funciones incorporadas:

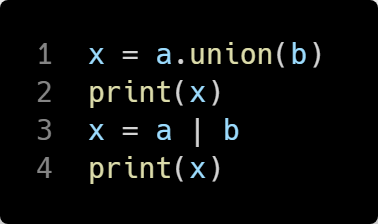
Dados los siguientes conjuntos, a y b:



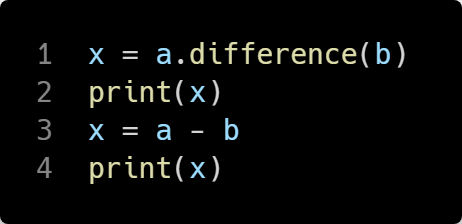
* intersection (elementos que están a la vez en a y en b)



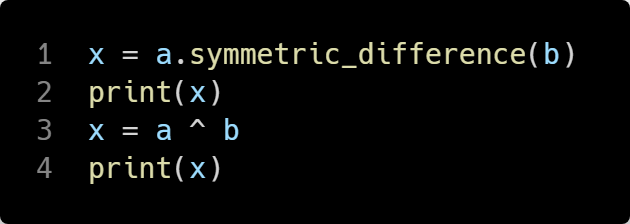
* union (elementos que están tanto en a como en b:)



* difference (elementos que están en a y no están en b)



* symmetric\_difference (elementos que están en a o en b, pero no en ambos conjuntos)



Otras funciones incorporadas que puedes probar:

* difference\_update
* intersection\_update
* symmetric\_difference\_update
* issuperset
* isdisjoint

| En Visual Studio Code puedes ver qué hace cada función de la siguiente forma:  - Escribe el nombre del conjunto  - Presiona el punto ' . '  - Presiona 'control + space'  - Con las teclas cursor, selecciona la función y presiona 'tab'  - Posiciona el cursor del mouse sobre el nombre de la función y verás la ayuda. |
| --- |

## Revisemos lo aprendido hasta aquí

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