Expresiones y Variables.

Las Expresiones describen un tipo de operación que ejecuta el ordenador. Las Expresiones son operaciones que ejecuta python, por ejemplo, operaciones aritméticas básicas como la suma de varios números.

En este caso, el resultado es 160. Los números son los operandos y los símbolos matemáticos, en este caso suma, son los operadores. Podemos realizar operaciones como la resta usando el signo de la resta. En este caso el resultado es un número negativo. Podemos realizar multiplicaciones usando el asterisco. El resultado es 25.

En este caso, los operandos son el menos y el asterisco. También podemos realizar la división

con la barra inclinada. 25 / 5 es 5,0. 25 / 6 es aproximadamente 4,167. En Python 3, la versión que usaremos durante este curso, ambos resultados son float. Podemos usar la doble barra para la división

entera, donde el resultado se redondea. Hay que tener en cuenta que, en algunos casos,

los resultados no son los mismos que en la división normal. Python sigue las convenciones matemáticas

cuando ejecuta expresiones matemáticas. Las siguientes operaciones están en un orden diferente. En ambos casos, Python ejecuta primero la multiplicación y luego la suma para obtener el resultado final. Hay muchas más operaciones que podéis hacer con Python, leed la documentación para ver más ejemplos. También veremos operaciones más complejas a lo largo del curso.

Las expresiones entre paréntesis se ejecutan primero. Luego, multiplicamos el resultado por 60. El resultado es 1.920. Ahora, echemos un vistazo a las variables. Podemos usar las variables para almacenar valores,

en este caso, asignamos el valor 1 a la variable my\_variable utilizando el operador de asignación, es decir, el signo igual. Entonces podemos utilizar el valor en otra parte del código escribiendo el nombre exacto de la variable. Vamos a utilizar dos puntos para indicar el valor de la variable. Podemos asignar un nuevo valor a my\_variable utilizando el operador de asignación. Le asignamos un valor de 10.

Ahora, la variable tiene un valor de 10. El valora antiguo de la variable no tiene importancia. Podemos almacenar los resultados de expresiones,

por ejemplo, añadir varios valores y asignar el resultado a x,

x ahora almacena el resultado. También podemos realizar operaciones con x y

guardar el resultado en una nueva variable y. y ahora tiene un valor de 2,666. También podemos realizar operaciones con x y

asignar el valor a x. La variable x, ahora tiene un valor 2,666. Como antes,

el antiguo valor de x no es importante. Podemos utilizar el comando type

en variables también. Es una buena práctica utilizar

nombres de variables significativos, para no tener que hacer un seguimiento

de lo que está haciendo la variable. Digamos que queremos

convertir el número de minutos a horas en los ejemplos señalados

en el siguiente conjunto de datos de música. Llamamos a la variable que contiene

el número total de minutos total\_min. Es común el uso del subrayado para

representar el comienzo de una nueva palabra. También se podría utilizar una letra mayúscula. Llamamos a la variable que contiene

el número total de horas, total\_hour. Podemos obtener el número total de

horas dividiendo total\_min por 60. El resultado es aproximadamente 2,367 horas. Si modificamos el valor

de la primera variable, se cambiará el valor de la variable. El resultado final cambiará en consecuencia, pero no tenemos que modificar

el resto del código.

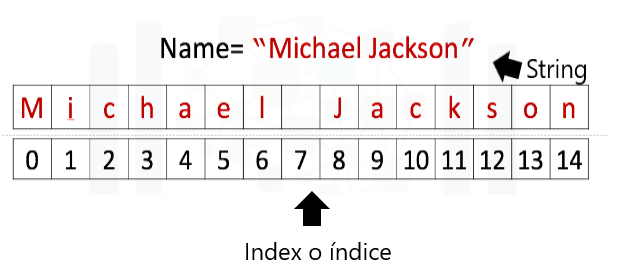
Cadenas o Strings

En Python, una cadena o string es una secuencia de caracteres.

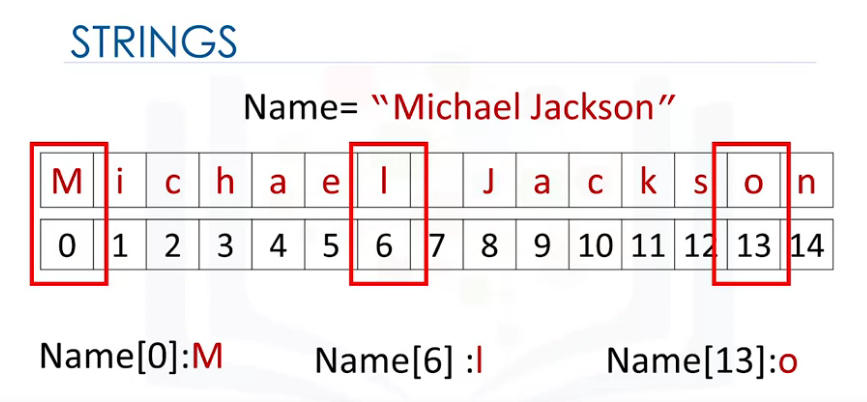
Una cadena se delimita mediante comillas dobles. También puede usar comillas simples. Una cadena puede ser espacios o números. Una cadena puede ser también caracteres especiales. Podemos enlazar o asignar una cadena a otra variable. Es útil pensar en una cadena como una secuencia ordenada.

“MICHAEL JACKSON” es un ejemplo de String o Cadena

Se puede acceder a cada elemento de la secuencia usando un índice representado por una matriz de números. El primer índice se puede acceder como sigue.



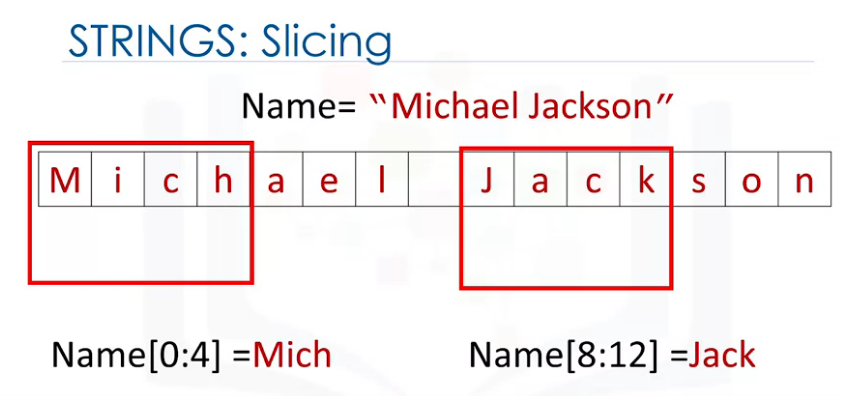
Podemos acceder al índice 6. Además, podemos acceder al índice 13.



También podemos usar indexación negativa con cadenas. El último elemento está dado por el índice menos uno. El primer elemento puede obtenerse por el índice menos 15 y así sucesivamente. Podemos asignar una cadena a otra variable.

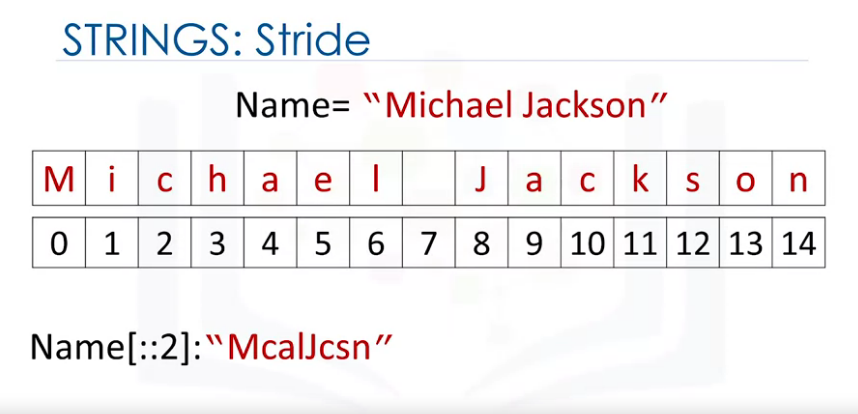


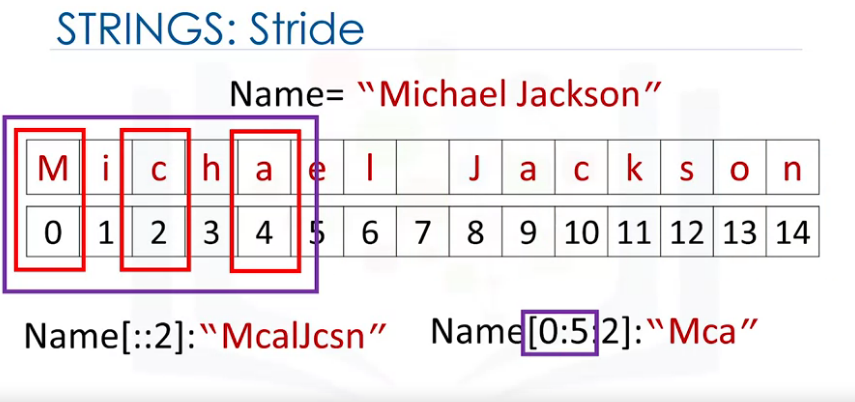
Es útil pensar en una cadena como una lista o tupla. Podemos tratar las cadenas como secuencias y ejecutar operaciones de secuencias.



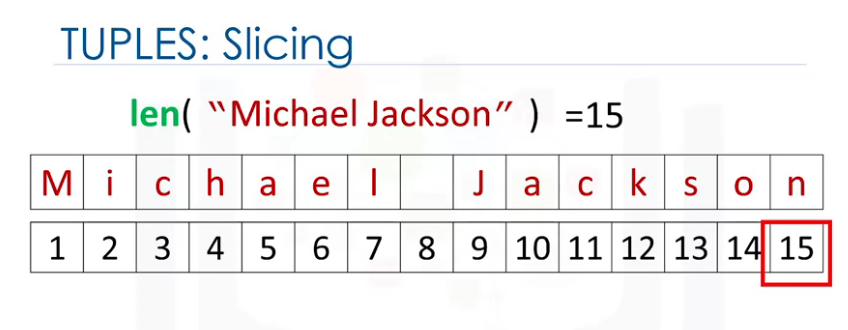
También podemos introducir un valor de paso como sigue. El dos indica que seleccionamos la variables

saltando de dos en dos.

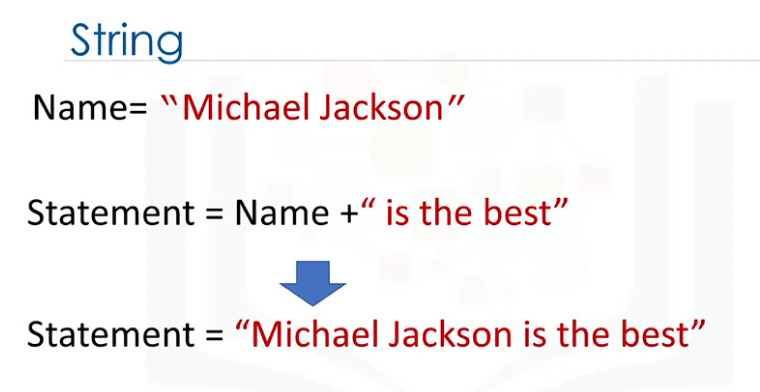
 También podemos seleccionar porciones. En este caso, seleccionamos de dos en dos hasta el índice cuatro.



Podemos utilizar el comando len para obtener la longitud de una cadena. Como hay 15 elementos, el resultado es 15.



Podemos concatenar o combinar cadenas. Utilizamos el símbolo de suma. El resultado es una nueva cadena que es una combinación de ambos.



Podemos replicar los valores de una cadena. Simplemente multiplicamos (\*) la cadena por el número de veces que queremos replicar, en este caso, tres.

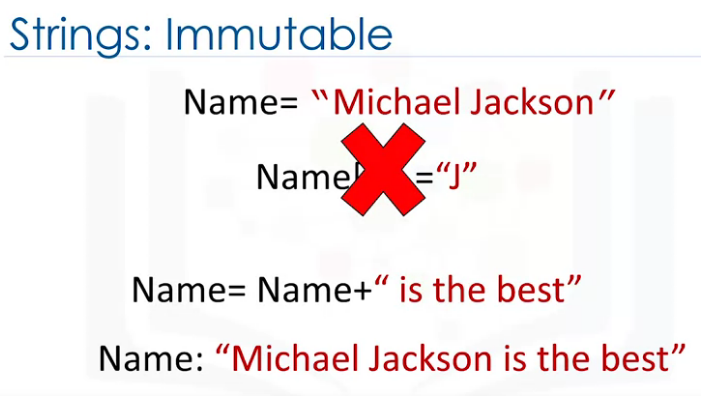
El resultado es una nueva cadena. La nueva cadena se compone de tres copias de la cadena original.



Esto significa que no se puede cambiar el valor de la cadena,

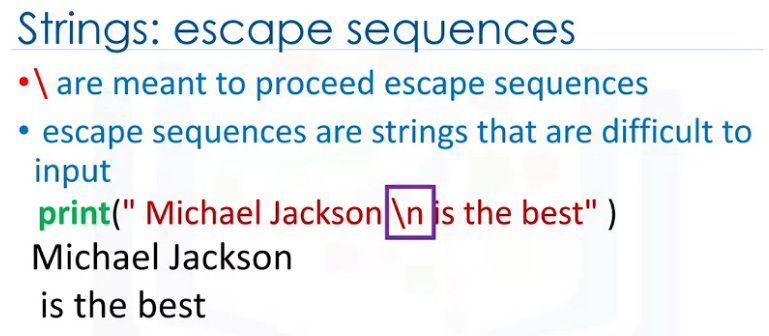
pero se puede crear una nueva cadena. Por ejemplo, puede crear una nueva cadena asignándole la variable original y concadenándola con una nueva cadena.

El resultado es una nueva cadena que cambia de 'Michael Jackson' a 'Michael Jackson is the best'.



Las cadenas son inmutables. La barra invertida representa el principio de una secuencia de escape. Las secuencias de escapes representan cadenas

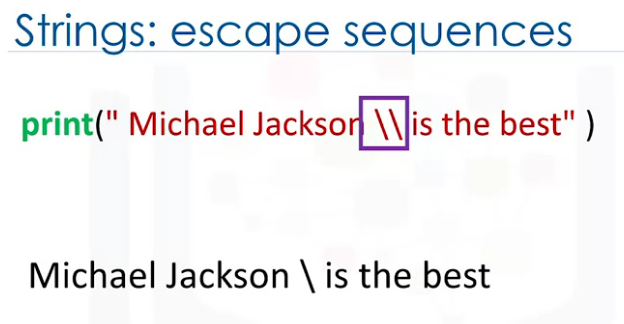
que pueden ser difíciles de escribir. Por ejemplo, \n representa una nueva línea. La salida es una nueva línea en el lugar de \n.



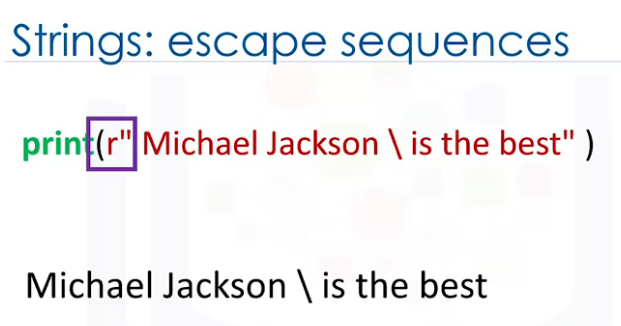
Del mismo modo, \t representa un tabulador. La salida es un tabulador en el lugar de \t.



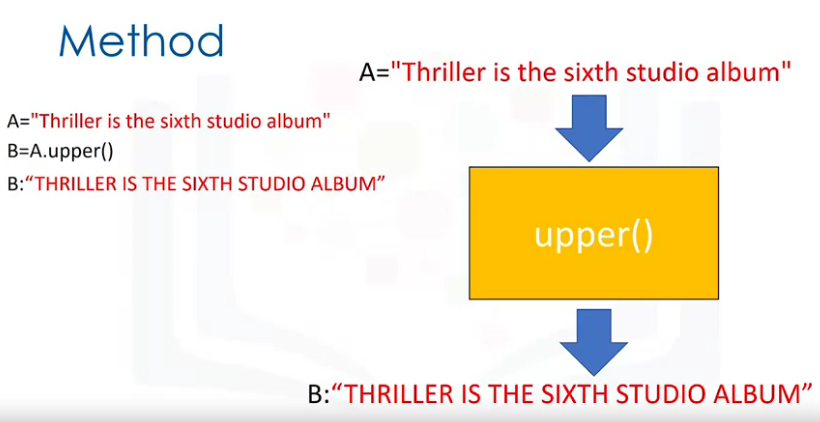
Si desea colocar una barra invertida en una cadena, utilice dos barras invertidas. El resultado es una barra invertida después de la secuencia de escape.



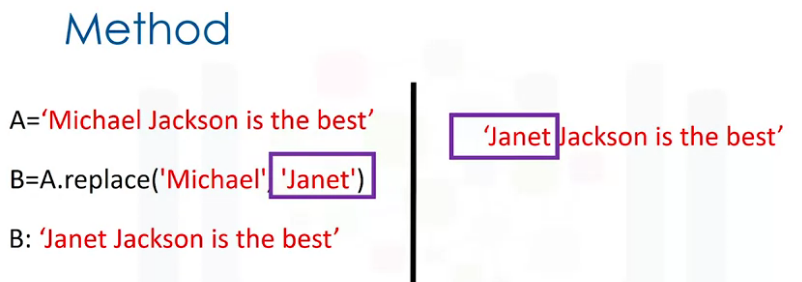
También podemos colocar una r delante de la cadena.



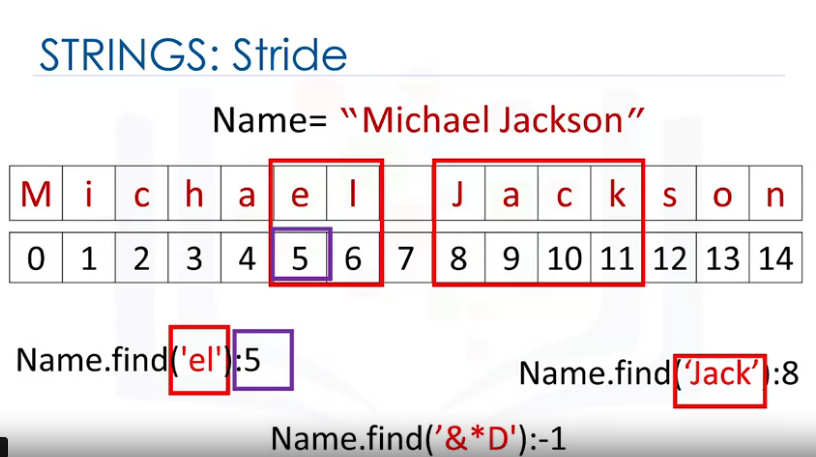
Ahora, echemos un vistazo a los métodos de cadena. Las cadenas son secuencias y como tal, tienen métodos que funcionan en listas y tuplas. Las cadenas también tienen un segundo conjunto de métodos que sólo funcionan en cadenas. Cuando aplicamos un método a la cadena A, obtenemos una nueva cadena B que es diferente de A. Vamos a hacer algunos ejemplos. Vamos a probar con el método upper. Este método convierte los caracteres en minúsculas a mayúsculas. En este ejemplo, asignamos a la variable A el valor siguiente. Aplicamos el método upper y lo asignamos a B. El valor de B es similar al de A, pero todos los caracteres son mayúsculas.



El método replace sustituye un segmento de la cadena, es decir, una subcadena, por una nueva cadena. Introducimos la parte de la cadena que queremos cambiar. El segundo argumento es el segmento por el que queremos cambiarla. El resultado es una nueva cadena con un segmento cambiado.

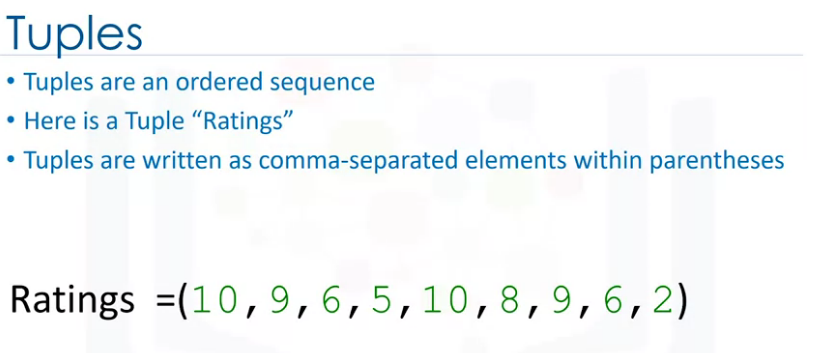


El método find busca subcadenas. El argumento es la subcadena que desea buscar. La salida es el primer índice de la subcadena. Podemos buscar la subcadena 'Jack'. Si la subcadena no está en la cadena, la salida es menos uno. Revise la documentación para ver más ejemplos.



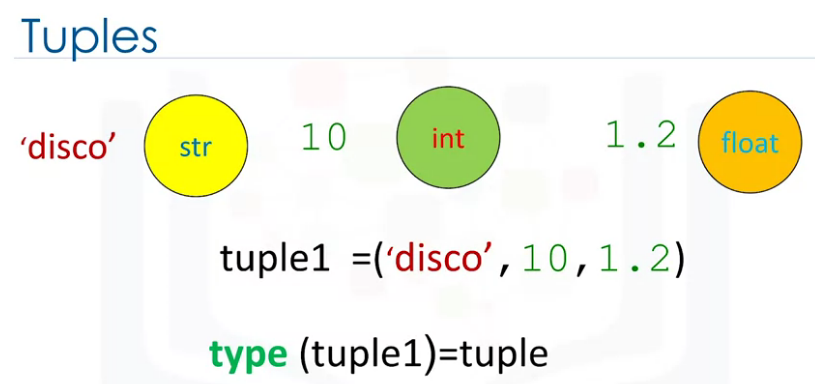
Listas y Tuplas.

Estos se denominan tipos de datos compuestos y son uno de los tipos de estructuras de datos más importantes en Python. Las tuplas son secuencias ordenadas. Aquí tenemos una tupla de puntuaciones. Las tuplas se expresan como elementos separados por coma entre paréntesis. Estos son los valores dentro de los paréntesis.



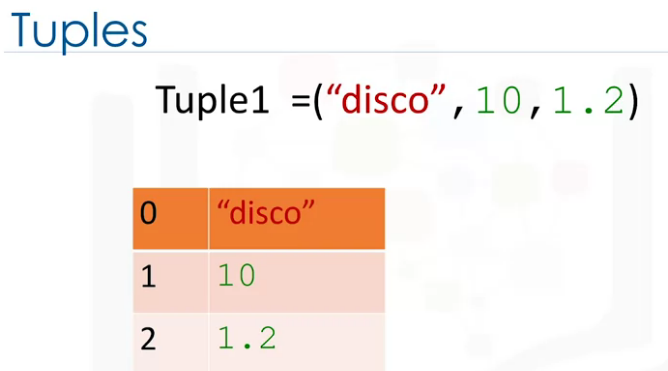
En Python, hay diferentes tipos, Strings, integer, Float.

Todos ellos pueden estar contenidos en una tupla. Pero el tipo de la variable es tupla.



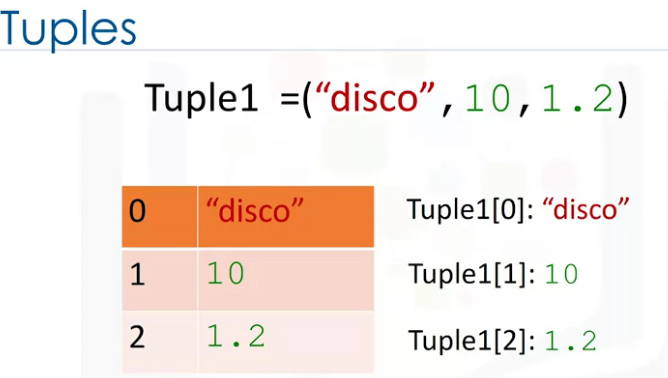
Cada elemento de una tupla puede ser accedido a través de un índice.

La siguiente tabla representa la relación entre el índice y los elementos de la tupla.



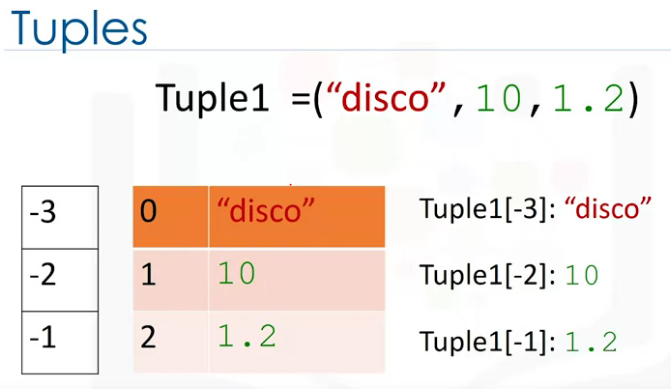
Se puede acceder al primer elemento por el nombre de la tupla, seguido de un corchete con

el número de índice, en este caso, 0. Podemos acceder al segundo elemento de la siguiente manera, también podemos acceder al último elemento.



En Python, podemos usar índice negativo. La relación es la siguiente,

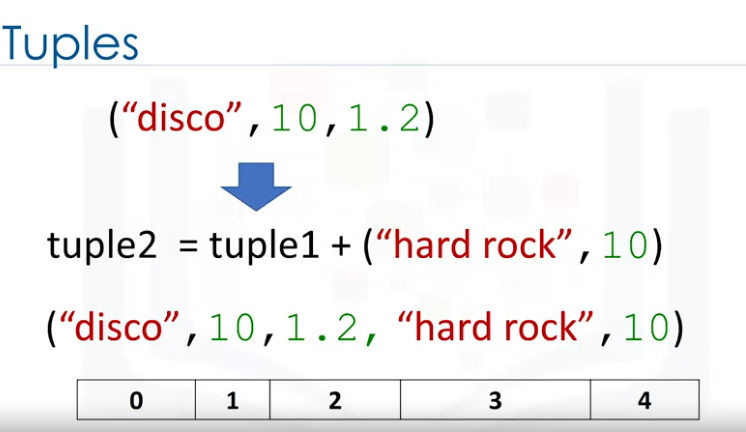
aquí se muestran los valores correspondientes.



Podemos concatenar o

combinar tuplas sumándolas, el resultado es el siguiente

con el siguiente índice.

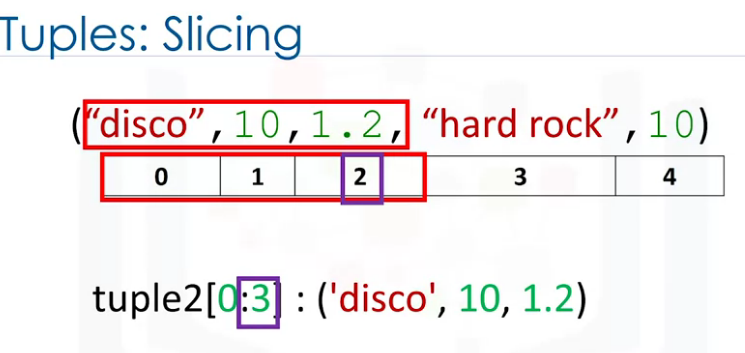


Si quisiéramos obtener varios elementos a partir

de una tupla, podríamos trocearla. Por ejemplo, si queremos los tres primeros

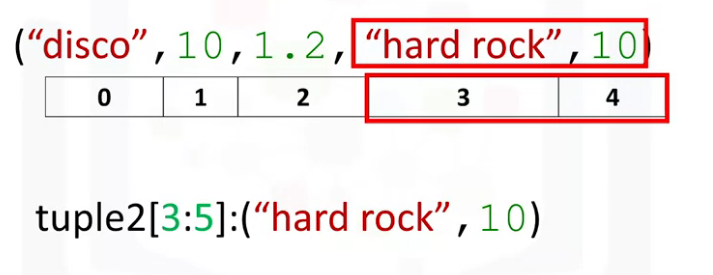
elementos, utilizamos el siguiente comando, el último índice es uno más

que el índice que quieres.



Del mismo modo, si queremos los dos últimos

elementos, usamos el siguiente comando.

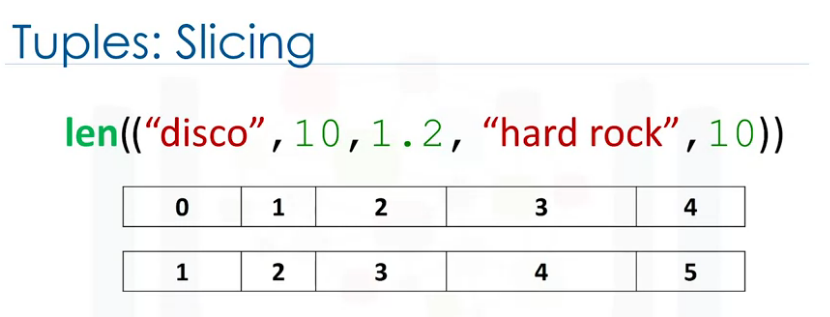


Observe cómo el último índice es uno

más que la longitud de la tupla, podemos usar el comando 'len' para

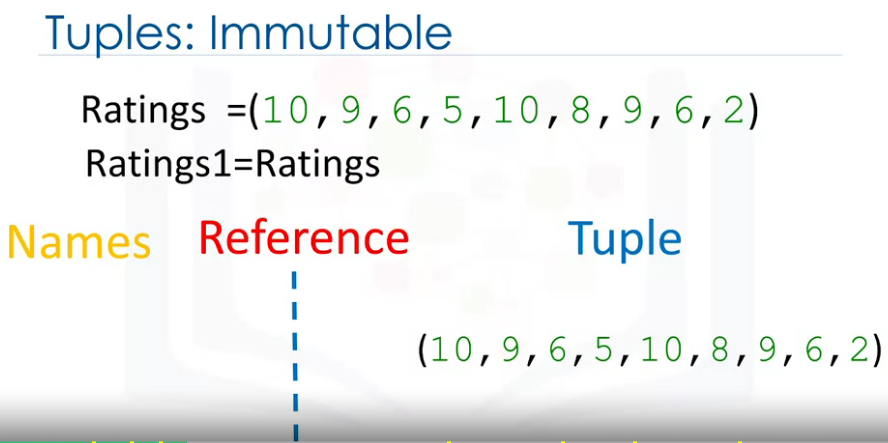
obtener la longitud de la tupla. Como hay cinco elementos,

el resultado es cinco.



Las tuplas son inmutables,

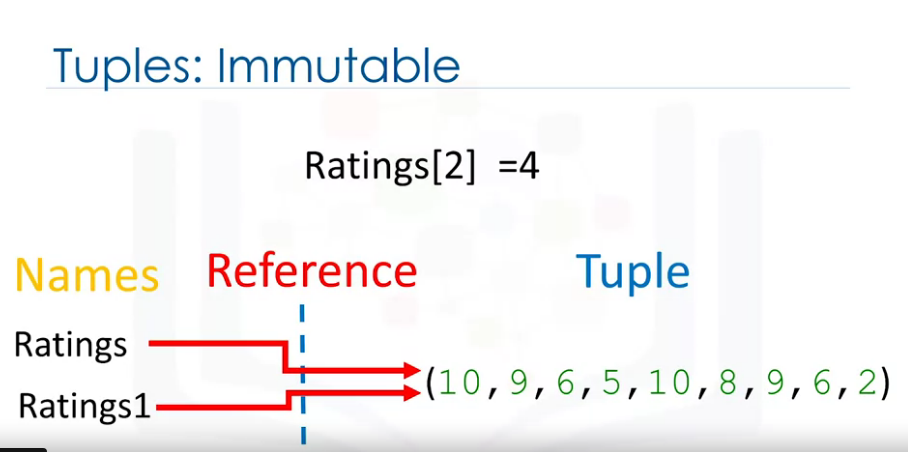
lo que significa que no podemos cambiarlas. Para entender por qué esto es importante, veamos qué sucede cuando fijamos la variable Ratings1 a Ratings.



Vamos a usar una imagen para proporcionar una

explicación simplificada de lo que está sucediendo. Cada variable no contiene una tupla sino que hace referencia al mismo

objeto tupla inmutable.



Consulte el módulo de objetos y clases para

saber más sobre objetos.

Digamos que queremos cambiar

el elemento en el índice 2. Debido a que las tuplas son inmutables,

no podemos, por lo tanto, Ratings1 no se verá afectada por un cambio en Ratings. Dado que la tupla es inmutable,

no podemos cambiarlo. Podemos asignar una tupla diferente

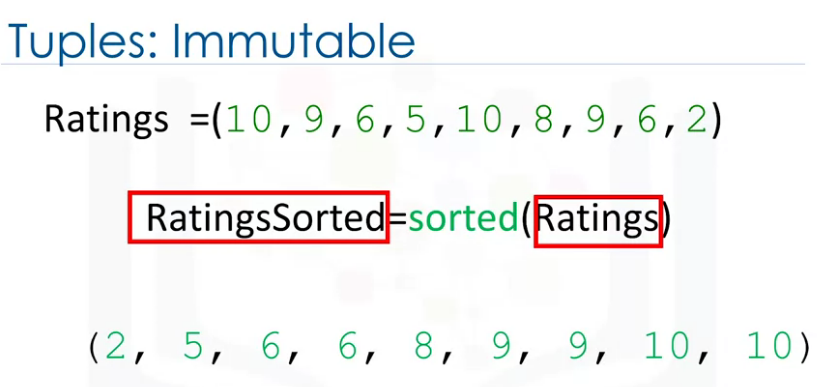
a la variable Ratings. Las variable Ratings ahora

hace referencia a otra tupla. Como consecuencia de la inmutabilidad,

si quisiéramos manipular una tupla, debemos crear una nueva tupla. Por ejemplo,

si queremos ordenar una tupla, utilizamos la función sorted,

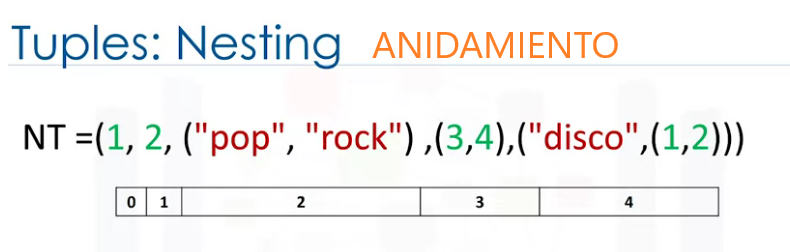
la entrada es la tupla original. La salida es una nueva tupla ordenada;



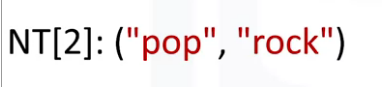
para conocer más sobre funciones, consulte nuestro vídeo de funciones.

Una tupla puede contener tanto otras tuplas como otros tipos de datos complejos, Esto se denomina anidamiento. Podemos acceder a estos elementos utilizando

los métodos estándar de indexación.



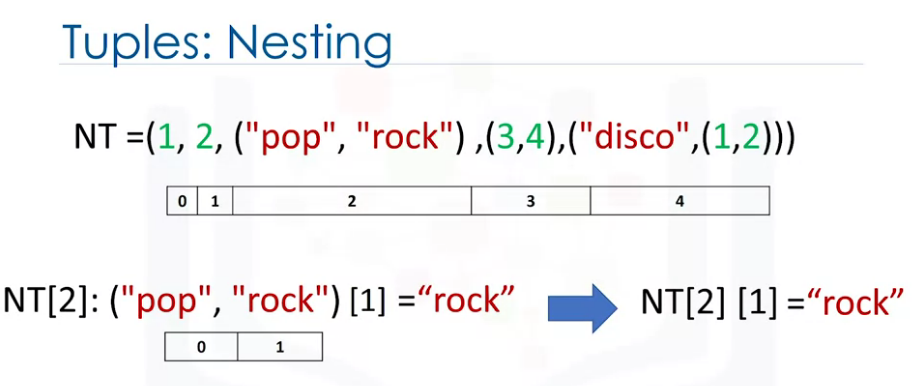
Si seleccionamos un índice con una tupla, se aplica la misma convención de índice.



Como tal

podemos acceder a los valores de la tupla. Por ejemplo, podríamos acceder al segundo elemento, podemos aplicar este indexado

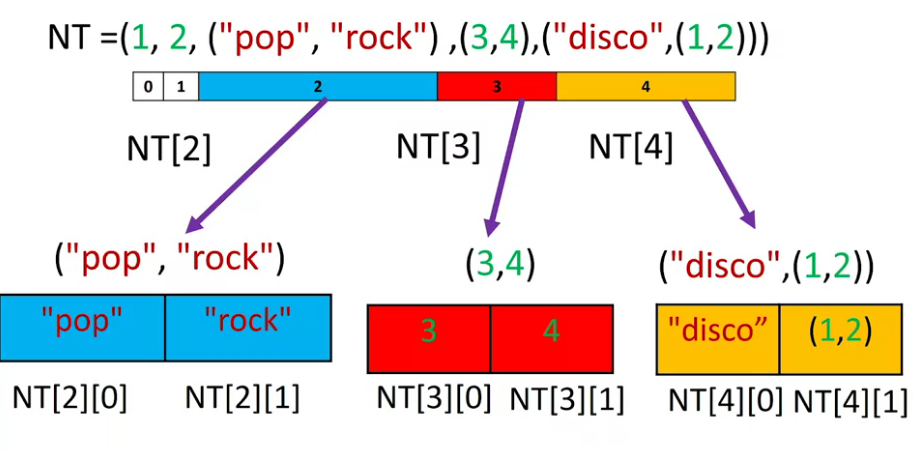
directamente a la variable tupla NT.



Es útil visualizar esto como un árbol. Podemos visualizar el anidamiento como un árbol. La tupla tiene los siguientes índices. Si consideramos los índices con otras tuplas, vemos que la tupla en el índice 2 contiene una tupla con dos elementos. Podemos acceder a esos dos índices. La misma regla aplica al índice 3. Podemos acceder también a los elementos

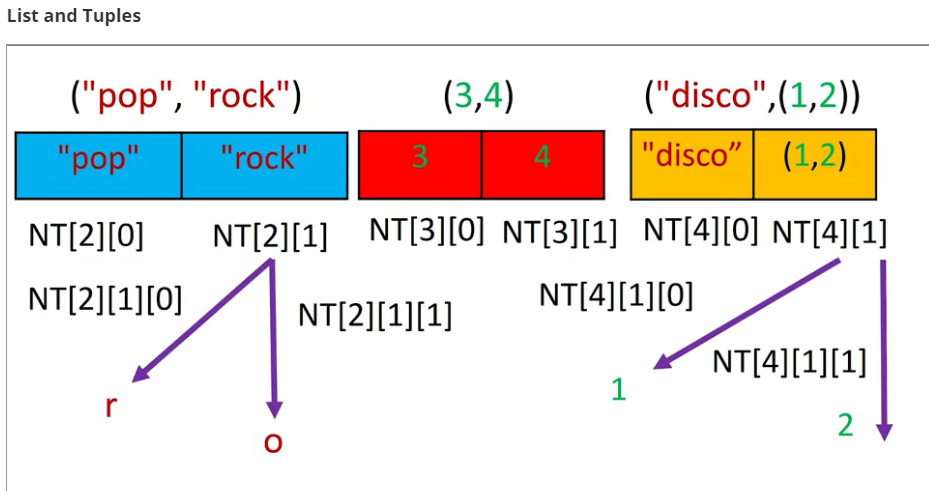
en esas tuplas. Podemos continuar el proceso, incluso podemos acceder a niveles más profundos del

árbol añadiendo otro corchete.



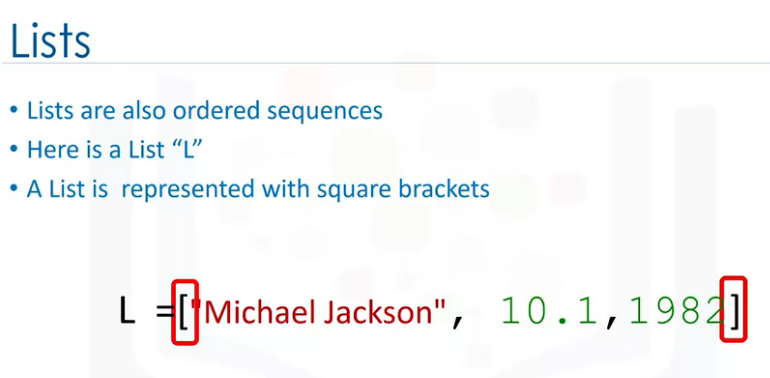
Podemos acceder a diferentes

caracteres de la cadena o a varios elementos de la segunda tupla contenida dentro de la primera.



Las listas son también una estructura de datos popular en Python. Las listas también son secuencias ordenadas. Aquí hay una lista L.

Una lista se representa con corchetes. En muchos aspectos, las listas son como tuplas. La principal diferencia es que son mutables.



Las listas pueden contener strings, floats, integers. Podemos anidar otras listas. También podemos anidar tuplas y otras estructuras de datos.

Las mismas convenciones de indexación se aplican para

el anidamiento. Al igual que las tuplas, cada elemento de una lista

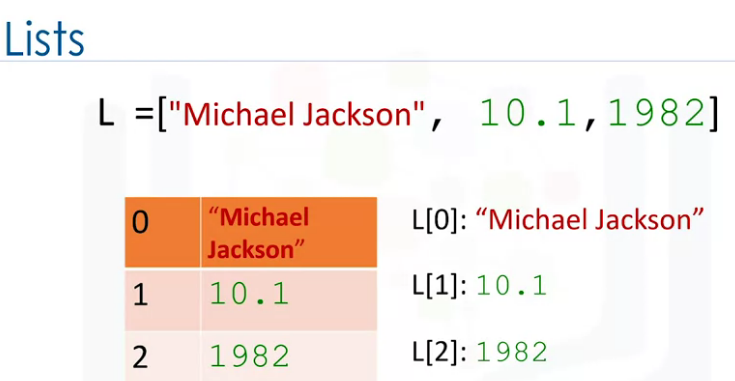
se puede acceder a través de un índice.

La siguiente tabla representa

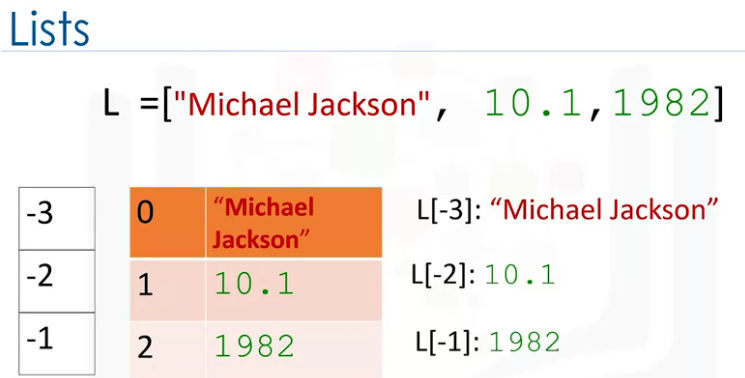
la relación entre el índice y los elementos de la lista. El primer elemento puede ser accedido por

el nombre de la lista seguida de un corchete con el número del índice, en este caso, 0. Podemos acceder al segundo

elemento de la siguiente manera. También podemos acceder al último elemento.



En Python, podemos usar un índice negativo. La relación es la siguiente. Los índices correspondientes son los siguientes.



También podemos trocear listas. Por ejemplo, si queremos los dos últimos

elementos de esta lista, usamos el siguiente comando.

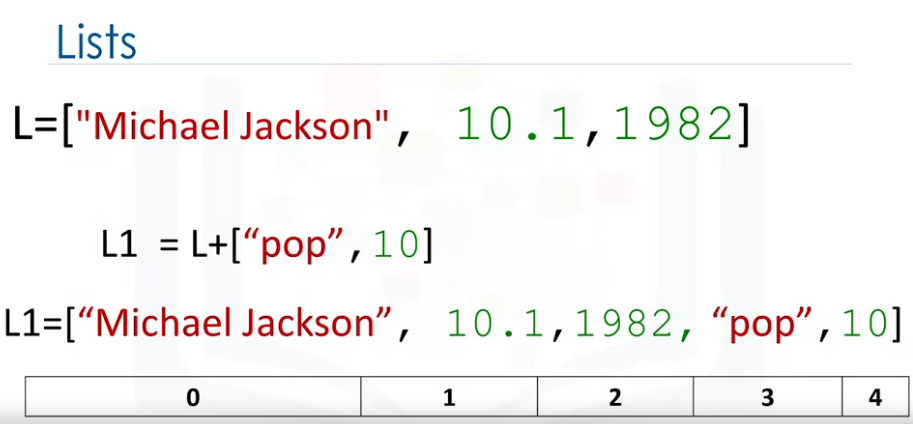


Observe cómo el último índice es uno

mayor que la longitud de la lista.

Las reglas de índice para listas y tuplas son iguales. Revise la documentación para ver más ejemplos.

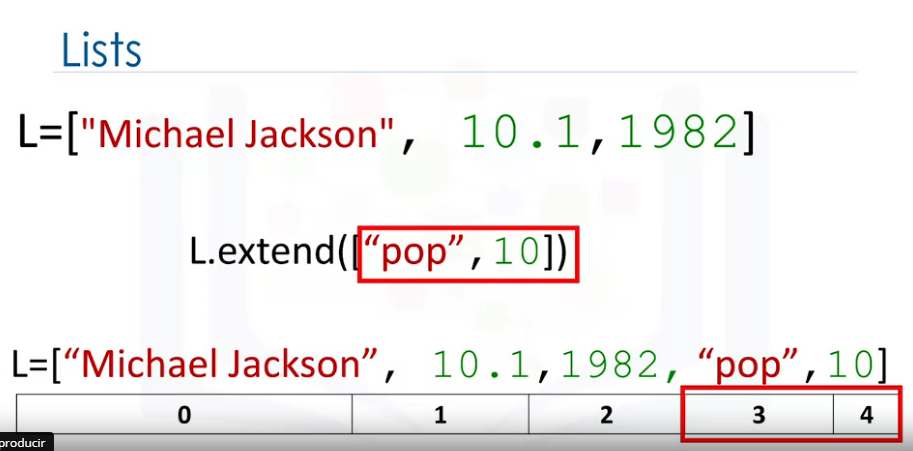
Podemos concatenar o combinar listas sumándolas. El resultado es el siguiente. La nueva lista tiene los siguientes índices.



Las listas son mutables, por lo tanto, podemos cambiarlas. Por ejemplo, aplicamos el método

'extend' añadiendo un punto seguido del nombre del método, y paréntesis. El argumento dentro de los paréntesis es una nueva lista que vamos a concatenar con la lista original. En este caso,

en lugar de crear una nueva lista L1, la lista original L se modifica añadiendo dos nuevos elementos.



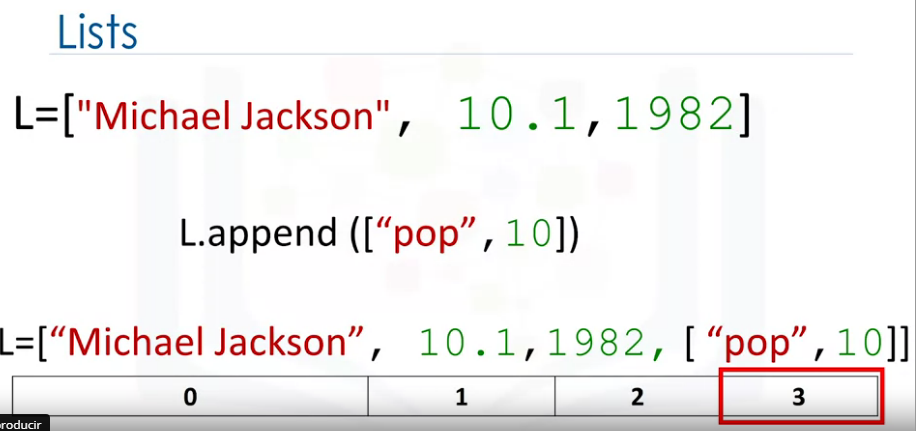
Para obtener más información sobre métodos, consulte nuestro video de objetos y clases.

Otro método similar es Append. Si aplicamos Append en lugar de Extend,

añadimos un elemento a la lista. Si miramos el índice, sólo hay un elemento más.

El índice 3 incluye la lista que hemos añadido. Cada vez que aplicamos un método,

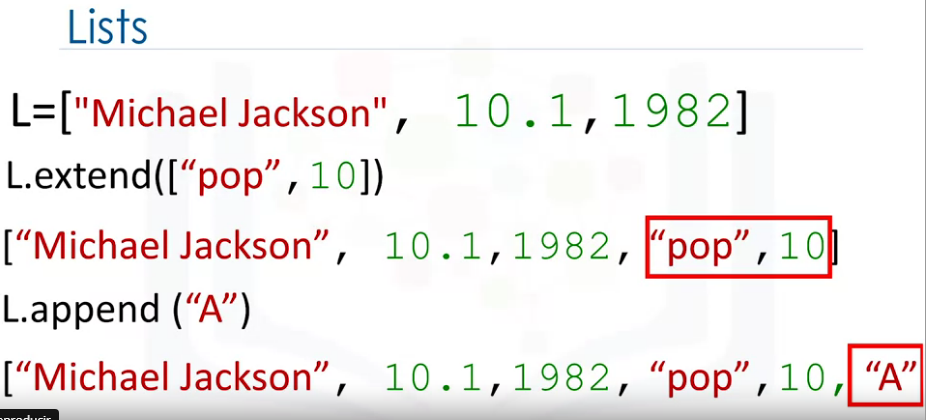
cambia la lista.



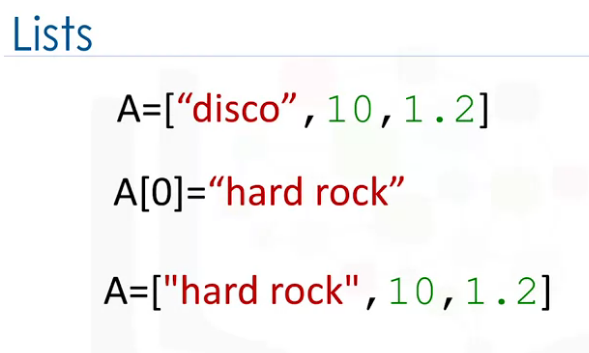
Si aplicamos Extend, añadimos dos nuevos elementos a la lista. La lista L se modifica

añadiendo dos nuevos elementos.

Si añadimos la cadena A, también cambia la lista, agregando la cadena A.



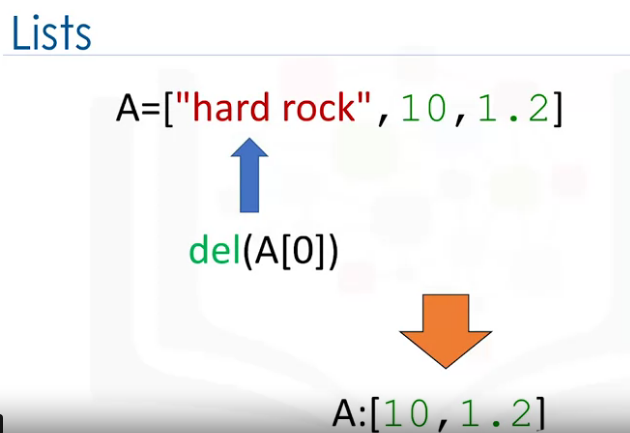
Como las listas son mutables, podemos cambiarlas. Por ejemplo, podemos cambiar el primer elemento de la siguiente manera.



La lista se convierte ahora en 'hard rock', 10, 1.2.

Podemos eliminar un elemento de una lista mediante el comando Del. Simplemente indicamos como argumento el elemento de la lista que nos gustaría eliminar .

Por ejemplo, si quisiéramos eliminar el primer elemento, el resultado se convierte en 10, 1.2. Podemos eliminar el segundo elemento. Esta operación elimina el segundo elemento de la lista.



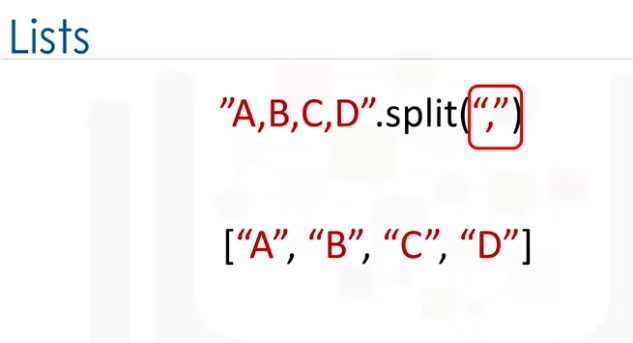
Podemos convertir una cadena en una lista utilizando Split. Por ejemplo,

el método Split convierte cada grupo de caracteres separados por espacios

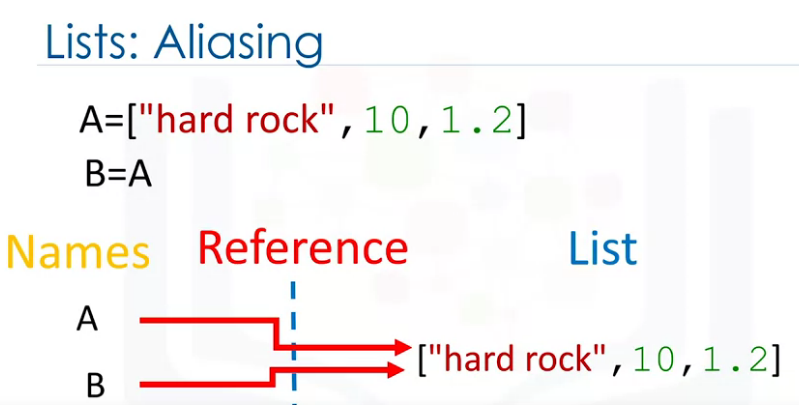
en un elemento de una lista.



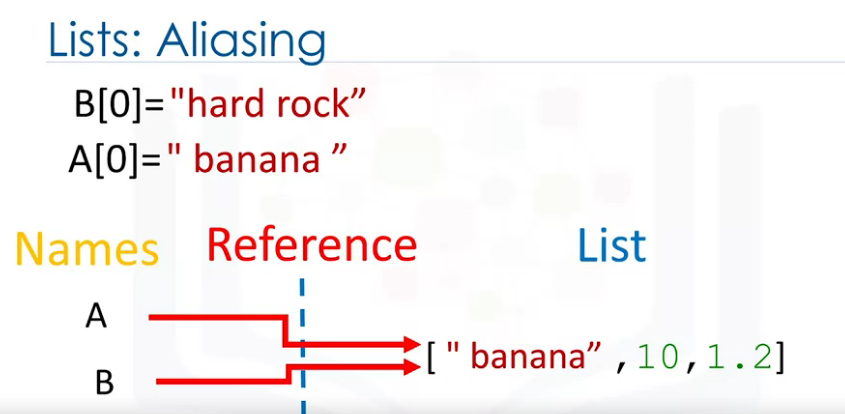
Podemos usar la función Split para separar cadenas por un carácter específico conocido como delimitador. Simplemente pasamos a Split como argumento el delimitador que queremos, en este caso, una coma. El resultado es una lista, cada elemento corresponde a un conjunto de caracteres que estaban separados por una coma.



Cuando se establece una variable B igual a A, tanto A como B hacen referencia a la misma lista.

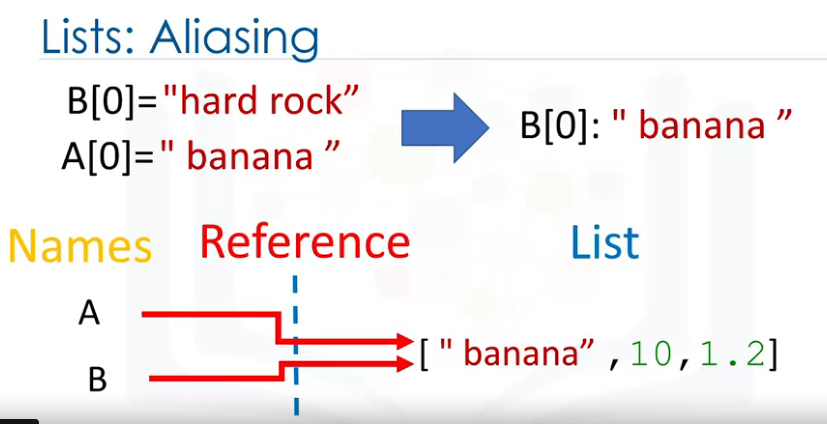


Varios nombres que se refieren al mismo objeto se conoce como aliasing. Sabemos por la presentación sobre listas que el primer elemento de B es 'hard rock'. Si cambiamos el primer elemento de A a 'banana', tenemos un efecto secundario. El valor de B cambiaría consecuentemente. A y B hacen referencia a la misma lista, por lo tanto si cambiamos A, la lista B también cambia.



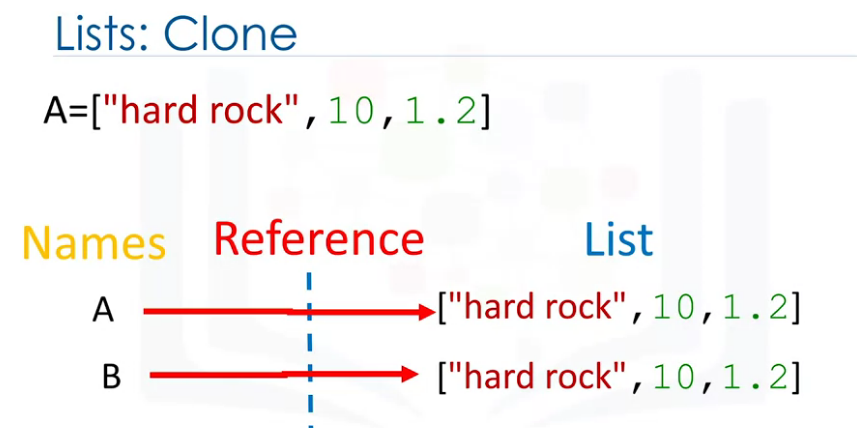
Si comprobamos el primer elemento

de B después de cambiar la lista A, tenemos 'banana' en vez de 'hard rock'.

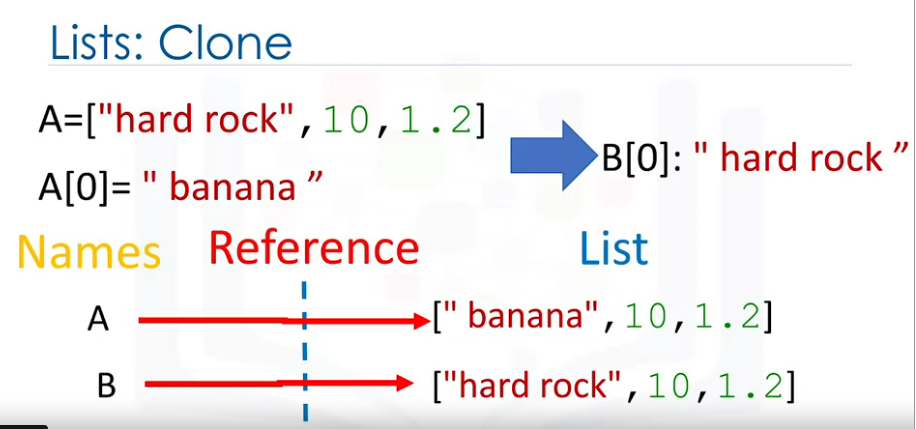


Puedes clonar la lista A

utilizando la siguiente sintaxis, la variable A hace referencia a una lista. La variable B hace referencia a una nueva copia opclon de la lista original.



Ahora, si cambias A, B no cambiará.



Podemos obtener más información sobre listas, tuplas y muchos otros objetos en Python

usando el comando 'help'. Simplemente pase como argumento la lista, tupla,

o cualquier otro objeto de Python. Consulte la documentación para saber más sobre listas.