LISTAS

DOCUMENTO INICIADO EL 30AGOS2021

**Listas de Python**

**List**

Las listas se utilizan para almacenar varios elementos en una sola variable.

Las listas son uno de los 4 tipos de datos integrados en Python que se utilizan para almacenar colecciones de datos, los otros 3 son Tuple, Set y Dictionary, todos con diferentes calidades y usos.

Las listas se crean utilizando corchetes:

### **Example**

Create a List:

thislist = ["apple", "banana", "cherry"]  
print(thislist)

**Elementos de la lista**

Los elementos de la lista están ordenados, se pueden cambiar y permiten valores duplicados.

Los elementos de la lista están indexados, el primer elemento tiene índice [0], el segundo elemento tiene índice [1], etc.

**Ordenado**

Cuando decimos que las listas están ordenadas, significa que los elementos tienen un orden definido y ese orden no cambiará.

Si agrega nuevos elementos a una lista, los nuevos elementos se colocarán al final de la lista.

**Nota**: Hay algunos métodos de lista que cambiarán el orden, pero en general: el orden de los elementos no cambiará.

**Cambiable**

La lista se puede cambiar, lo que significa que podemos cambiar, agregar y eliminar elementos en una lista después de que se haya creado.

**Permitir duplicados**

Dado que las listas están indexadas, las listas pueden tener elementos con el mismo valor:

### **Example**

Lists allow duplicate values:

thislist = ["apple", "banana", "cherry", "apple", "cherry"]  
print(thislist)

**Longitud de lista**

Para determinar cuántos elementos tiene una lista, use la función len():

### **Example**

Print the number of items in the list:

thislist = ["apple", "banana", "cherry"]  
print(len(thislist))

**Elementos de lista: tipos de datos**

Los elementos de la lista pueden ser de cualquier tipo de datos:

### **Example**

String, int and boolean data types:

list1 = ["apple", "banana", "cherry"]  
list2 = [1, 5, 7, 9, 3]  
list3 = [True, False, False]

Una lista puede contener diferentes tipos de datos:

### **Example**

A list with strings, integers and boolean values:

list1 = ["abc", 34, True, 40, "male"]

## type()

Desde la perspectiva de Python, las listas se definen como objetos con el tipo de datos 'list':

<class 'list'>

### **Example**

What is the data type of a list?

mylist = ["apple", "banana", "cherry"]  
print(type(mylist))

**El constructor list()**

También es posible utilizar el constructor list() al crear una nueva lista.

### **Example**

Using the list() constructor to make a List:

thislist = list(("apple", "banana", "cherry")) # note the double round-brackets  
print(thislist)

**Colecciones de Python (Arrays)**

Aquí hay cuatro tipos de datos de colecciones en el lenguaje de programación Python:

* **List** es una colección ordenada y modificable. Permite miembros duplicados.
* **Tuple** es una colección ordenada e inmutable. Permite miembros duplicados.
* **Set** es una colección desordenada y no indexada. No hay miembros duplicados.
* **Dictionary** es una colección ordenada\* y modificable. No hay miembros duplicados.

\* A partir de la versión 3.7 de Python, los diccionarios están ordenados. En Python 3.6 y versiones anteriores, los diccionarios están desordenados.

Al elegir un tipo de colección, es útil comprender las propiedades de ese tipo. Elegir el tipo correcto para un conjunto de datos en particular podría significar la retención del significado y podría significar un aumento en la eficiencia o la seguridad.

**Python: Acceso a los elementos de la lista**

**Acceso a los Elementos**

Los elementos de la lista están indexados y puede acceder a ellos refiriéndose al número de índice:

### **Example**

Print the second item of the list:

thislist = ["apple", "banana", "cherry"]  
print(thislist[1])

**Nota**: El primer elemento tiene índice 0.

**Indexación negativa**

La indexación negativa significa comenzar desde el final -1 se refiere al último elemento, -2 se refiere al penúltimo elemento, etc.

### **Example**

Print the last item of the list:

thislist = ["apple", "banana", "cherry"]  
print(thislist[-1])

**Rango de índices**

Puede especificar un rango de índices especificando dónde comenzar y dónde terminar el rango.

Al especificar un rango, el valor de retorno será una nueva lista con los elementos especificados.

### **Example**

Return the third, fourth, and fifth item:

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[2:5])

**Nota:** La búsqueda comenzará en el índice 2 (incluido) y terminará en el índice 5 (no incluido).

Recuerde que el primer elemento tiene índice 0.

Al omitir el valor inicial, el rango comenzará en el primer elemento:

### **Example**

This example returns the items from the beginning to, but NOT including, "kiwi":

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[:4])

Al omitir el valor final, el rango continuará hasta el final de la lista:

### **Example**

This example returns the items from "cherry" to the end:

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[2:])

**Rango de índices negativos**

Especifique índices negativos si desea iniciar la búsqueda desde el final de la lista:

### **Example**

This example returns the items from "orange" (-4) to, but NOT including "mango" (-1):

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[-4:-1])

**Compruebe si el elemento existe**

Para determinar si un elemento específico está presente en una lista, use la palabra clave in:

### **Example**

Check if "apple" is present in the list:

thislist = ["apple", "banana", "cherry"]  
if "apple" in thislist:  
  print("Yes, 'apple' is in the fruits list")

**Python - Cambiar elementos de la lista**

**Cambiar el valor del elemento**

Para cambiar el valor de un elemento específico, consulte el número de índice:

### **Example**

Change the second item:

thislist = ["apple", "banana", "cherry"]  
thislist[1] = "blackcurrant"  
print(thislist)

**Cambiar un rango de valores del elemento**

Para cambiar el valor de los elementos dentro de un rango específico, defina una lista con los nuevos valores y consulte el rango de números de índice donde desea insertar los nuevos valores:

### **Example**

Change the values "banana" and "cherry" with the values "blackcurrant" and "watermelon":

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]  
thislist[1:3] = ["blackcurrant", "watermelon"]  
print(thislist)

Si inserta más elementos de los que reemplaza, los elementos nuevos se insertarán donde haya especificado y los elementos restantes se moverán en consecuencia:

### **Example**

Change the second value by replacing it with two new values:

thislist = ["apple", "banana", "cherry"]  
thislist[1:2] = ["blackcurrant", "watermelon"]  
print(thislist)

**Nota**: La longitud de la lista cambiará cuando el número de elementos insertados no coincida con el número de elementos reemplazados.

Si inserta menos elementos de los que reemplaza, los elementos nuevos se insertarán donde haya especificado y los elementos restantes se moverán en consecuencia:

### **Example**

Change the second and third value by replacing it with one value:

thislist = ["apple", "banana", "cherry"]  
thislist[1:3] = ["watermelon"]  
print(thislist)

**Insertar elementos**

Para insertar un nuevo elemento de la lista, sin reemplazar ninguno de los valores existentes, podemos usar el método insert().

El método insert() inserta un elemento en el índice especificado:

### **Example**

Insert "watermelon" as the third item:

thislist = ["apple", "banana", "cherry"]  
thislist.insert(2, "watermelon")  
print(thislist)

**Nota**: Como resultado del ejemplo anterior, la lista ahora contendrá 4 elementos.

**Python - Agregar elementos de lista**

**Agregar elementos**

Para agregar un elemento al final de la lista, use el método append():

### **Example**

Using the append() method to append an item:

thislist = ["apple", "banana", "cherry"]  
thislist.append("orange")  
print(thislist)

**Insertar elementos**

Para insertar un elemento de la lista en un índice específico, use el método insert().

El método insert() inserta un elemento en el índice especificado:

### **Example**

Insert an item as the second position:

thislist = ["apple", "banana", "cherry"]  
thislist.insert(1, "orange")  
print(thislist)

**Nota**: Como resultado de los ejemplos anteriores, las listas ahora contendrán 4 elementos.

**Ampliar lista**

Para agregar elementos de otra lista a la lista actual, use el método extend().

### **Example**

Add the elements of tropical to thislist:

thislist = ["apple", "banana", "cherry"]  
tropical = ["mango", "pineapple", "papaya"]  
thislist.extend(tropical)  
print(thislist)

Los elementos se agregarán al final de la lista.

**Agregar cualquier iterable**

El método extend() no tiene que agregar listas, puede agregar cualquier objeto iterable (tuplas, conjuntos, diccionarios, etc.).

### **Example**

Add elements of a tuple to a list:

thislist = ["apple", "banana", "cherry"]  
thistuple = ("kiwi", "orange")  
thislist.extend(thistuple)  
print(thislist)

**Python - Eliminar elementos de la lista**

**Eliminar elemento especificado**

El método remove() elimina el elemento especificado.

### **Example**

Remove "banana":

thislist = ["apple", "banana", "cherry"]  
thislist.remove("banana")  
print(thislist)

**Eliminar índice especificado**

El método pop() elimina el índice especificado.

Si no especifica el índice, el método pop() elimina el último elemento.

### **Example**

Remove the last item:

thislist = ["apple", "banana", "cherry"]  
thislist.pop()  
print(thislist)

La palabra clave del también elimina el índice especificado:

### **Example**

Remove the first item:

thislist = ["apple", "banana", "cherry"]  
del thislist[0]  
print(thislist)

La palabra clave del también puede eliminar la lista por completo.

### **Example**

Delete the entire list:

thislist = ["apple", "banana", "cherry"]  
del thislist

**Limpiar la lista**

El método clear() vacía la lista.

La lista aún permanece, pero no tiene contenido.

### **Example**

Clear the list content:

thislist = ["apple", "banana", "cherry"]  
thislist.clear()  
print(thislist)

CONTINUACION DEL DIA 31AGO2021

**Python - Listas de bucles**

**Recorrer una lista**

Puede recorrer los elementos de la lista utilizando un bucle for:

### **Example**

Print all items in the list, one by one:

thislist = ["apple", "banana", "cherry"]  
for x in thislist:  
  print(x)

Obtenga más información sobre los bucles for en nuestro capítulo de bucles For de Python.

**Recorrer los números de índice**

También puede recorrer los elementos de la lista consultando su número de índice.

Utilice las funciones range() y len() para crear un iterable adecuado.

### **Example**

Print all items by referring to their index number:

thislist = ["apple", "banana", "cherry"]  
for i in range(len(thislist)):  
  print(thislist[i])

El iterable creado en el ejemplo anterior es [0, 1, 2].

**Usar un bucle while**

Puede recorrer los elementos de la lista utilizando un ciclo while.

Use la función len() para determinar la longitud de la lista, luego comience en 0 y recorra los elementos de la lista consultando sus índices.

Recuerde aumentar el índice en 1 después de cada iteración.

### **Example**

Print all items, using a while loop to go through all the index numbers

thislist = ["apple", "banana", "cherry"]  
i = 0  
while i < len(thislist):  
  print(thislist[i])  
  i = i + 1

Obtenga más información sobre los bucles while en nuestro capítulo de bucles while de Python.

**Bucle mediante la comprensión de listas**

List Comprehension ofrece la sintaxis más corta para recorrer listas:

### **Example**

A short hand for loop that will print all items in a list:

thislist = ["apple", "banana", "cherry"]  
[print(x) for x in thislist]

Obtenga más información sobre la comprensión de listas en el siguiente capítulo: Comprensión de listas.

# **Python - Comprensión de listas (List Comprehension)**

**Comprensión de listas**

La comprensión de listas ofrece una sintaxis más corta cuando desea crear una nueva lista basada en los valores de una lista existente.

Ejemplo:

Basándose en una lista de frutas, desea una nueva lista que contenga solo las frutas con la letra "a" en el nombre.

Sin la comprensión de la lista, tendrá que escribir una declaración for con una prueba condicional dentro:

### **Example**

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
newlist = []  
  
for x in fruits:  
  if "a" in x:  
    newlist.append(x)  
  
print(newlist)

Con la comprensión de listas, puede hacer todo eso con solo una línea de código:

### **Example**

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
  
newlist = [x for x in fruits if "a" in x]  
  
print(newlist)

**La sintaxis**

newlist = [expression for item in iterable if condition == True]

El valor de retorno es una nueva lista, dejando la lista anterior sin cambios.

**Condición**

La condición es como un filtro que solo acepta los elementos que se valoran como True.

### **Example**

Only accept items that are not "apple":

newlist = [x for x in fruits if x != "apple"]

La condición if x! = "Apple" devolverá True para todos los elementos que no sean "apple", haciendo que la nueva lista contenga todas las frutas excepto "apple".

La condición es opcional y se puede omitir:

### **Example**

With no if statement:

newlist = [x for x in fruits]

**Iterable**

El iterable puede ser cualquier objeto iterable, como una lista, tupla, conjunto, etc.

El iterable puede ser cualquier objeto iterable, como una lista, tupla, conjunto (set), etc.

### **Example**

You can use the range() function to create an iterable:

newlist = [x for x in range(10)]

Mismo ejemplo, pero con una condición:

### **Example**

Accept only numbers lower than 5:

newlist = [x for x in range(10) if x < 5]

**Expresión**

La expresión es el elemento actual en la iteración, pero también es el resultado, que puede manipular antes de que termine como un elemento de la lista en la nueva lista:

### **Example**

Set the values in the new list to upper case:

newlist = [x.upper() for x in fruits]

Puede configurar el resultado como desee:

### **Example**

Set all values in the new list to 'hello':

newlist = ['hello' for x in fruits]

La expresión también puede contener condiciones, no como un filtro, sino como una forma de manipular el resultado:

### **Example**

Return "orange" instead of "banana":

newlist = [x if x != "banana" else "orange" for x in fruits]

La expresión del ejemplo anterior dice:

*"Devuelve el elemento si no es banana, si es banana devuelve naranja".*

Continuación el 04 de sept 2021, sábado

**Python: ordenar listas**

**Ordenar lista alfanuméricamente**

Los objetos de lista tienen un método sort() que ordenará la lista alfanuméricamente, de forma ascendente, por defecto:

### **Example**

Sort the list alphabetically:

thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]  
thislist.sort()  
print(thislist)

### **Example**

Sort the list numerically:

thislist = [100, 50, 65, 82, 23]  
thislist.sort()  
print(thislist)

**Orden descendiente**

Para ordenar de forma descendente, use el argumento de palabra clave reverse = True:

### **Example**

Sort the list descending:

thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]  
thislist.sort(reverse = True)  
print(thislist)

### **Example**

Sort the list descending:

thislist = [100, 50, 65, 82, 23]  
thislist.sort(reverse = True)  
print(thislist)

**Personalizar la función de ordenar**

También puede personalizar su propia función utilizando el argumento de palabra clave key = function.

La función devolverá un número que se utilizará para ordenar la lista (el número más bajo primero):

### **Example**

Ordene la lista según lo cerca que esté el número de 50:

def myfunc(n):  
  return abs(n - 50)  
  
thislist = [100, 50, 65, 82, 23]  
thislist.sort(key = myfunc)  
print(thislist)

**Orden que no distingue entre mayúsculas y minúsculas**

De forma predeterminada, el método sort() distingue entre mayúsculas y minúsculas, lo que hace que todas las letras mayúsculas se ordenen antes que las minúsculas:

### **Example**

Case La clasificación sensible a mayúsculas y minúsculas puede dar un resultado inesperado:

thislist = ["banana", "Orange", "Kiwi", "cherry"]  
thislist.sort()  
print(thislist)

Afortunadamente, podemos usar funciones integradas como funciones clave al ordenar una lista.

Entonces, si desea una función de ordenación que no distinga entre mayúsculas y minúsculas, use str.lower como una función clave:

### **Example**

Realizar una clasificación de la lista que no distinga entre mayúsculas y minúsculas:

thislist = ["banana", "Orange", "Kiwi", "cherry"]  
thislist.sort(key = str.lower)  
print(thislist)

**Orden inverso**

¿Qué sucede si desea invertir el orden de una lista, independientemente del alfabeto?

El método reverse() invierte el orden de clasificación actual de los elementos.

### **Example**

Reverse the order of the list items:

thislist = ["banana", "Orange", "Kiwi", "cherry"]  
thislist.reverse()  
print(thislist)

**Python - Copiar listas**

**Copiar una lista**

No puede copiar una lista simplemente escribiendo list2 = list1, porque: list2 solo será una referencia a list1, y los cambios realizados en list1 también se realizarán automáticamente en list2.

Hay formas de hacer una copia, una forma es usar el método de lista incorporado copy().

### **Example**

Make a copy of a list with the copy() method:

thislist = ["apple", "banana", "cherry"]  
mylist = thislist.copy()  
print(mylist)

Otra forma de hacer una copia es usar el método incorporado list().

### **Example**

Make a copy of a list with the list() method:

thislist = ["apple", "banana", "cherry"]  
mylist = list(thislist)  
print(mylist)

**Python: unir lista**

**Unir dos listas**

Hay varias formas de unir o concatenar dos o más listas en Python.

Una de las formas más sencillas es utilizar el operador +.

### **Example**

Join two list:

list1 = ["a", "b", "c"]  
list2 = [1, 2, 3]  
  
list3 = list1 + list2  
print(list3)

Otra forma de unir dos listas es agregando todos los elementos de list2 a list1, uno por uno:

### **Example**

Append list2 into list1:

list1 = ["a", "b" , "c"]  
list2 = [1, 2, 3]  
  
for x in list2:  
  list1.append(x)  
  
print(list1)

O puede usar el método extend(), cuyo propósito es agregar elementos de una lista a otra lista:

### **Example**

Use the extend() method to add list2 at the end of list1:

list1 = ["a", "b" , "c"]  
list2 = [1, 2, 3]  
  
list1.extend(list2)  
print(list1)

**Python - Métodos de Listas/Arreglos**

**Métodos de lista**

Python tiene un conjunto de métodos integrados que puede usar en listas.

Python has a set of built-in methods that you can use on lists.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Agrega un elemento al final de la lista. |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Elimina todos los elementos de la lista. |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Devuelve una copia de la lista. |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Devuelve el número de elementos con el valor especificado. |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Agregue los elementos de una lista (o cualquier iterable), al final de la lista  actual |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Devuelve el índice del primer elemento con el valor especificado. |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Agrega un elemento en la posición especificada |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Elimina el elemento en la posición especificada |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Elimina el elemento con el valor especificado |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Invierte el orden de la lista |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Ordena la lista |

Actividad iniciada el 9 de sept de 2021

**Python - Métodos de lista**

**Método append() de lista de Python**

### **Example**

Add an element to the fruits list:

fruits = ['apple', 'banana', 'cherry']  
fruits.append("orange")

**Definición y uso**

El método append() agrega un elemento al final de la lista.

**Sintaxis**

list.append(elmnt)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| elmnt | Requerido. Un elemento de cualquier tipo (cadena, número, objeto,  etc.) |

### **Más ejemplos**

### **Example**

Add a list to a list:

a = ["apple", "banana", "cherry"]  
b = ["Ford", "BMW", "Volvo"]  
a.append(b)

**Método Python List clear()**

### **Example**

Remove all elements from the fruits list:

fruits = ['apple', 'banana', 'cherry', 'orange']  
  
fruits.clear()

**Definición y uso**

El método clear() elimina todos los elementos de una lista.

**Sintaxis**

list.clear()

**Valores paramétricos**

Sin parámetros

**Método list copy() de Python**

### **Example**

Copy the fruits list:

fruits = ['apple', 'banana', 'cherry', 'orange']  
  
x = fruits.copy()

**Definición y uso**

El método copy() devuelve una copia de la lista especificada.

**Sintaxis**

list.copy()

**Valores paramétricos**

Sin parámetros

**Método Python List count ()**

### **Example**

Return the number of times the value "cherry" appears in the fruits list:

fruits = ['apple', 'banana', 'cherry']  
  
x = fruits.count("cherry")

**Definición y uso**

El método count() devuelve el número de elementos con el valor especificado.

**Sintaxis**

list.count(value)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| value | Requerido. Cualquier tipo (cadena, número, lista, tupla, etc.).  El valor a buscar. |

**Más ejemplos**

### **Example**

Return the number of times the value 9 appears int the list:

points = [1, 4, 2, 9, 7, 8, 9, 3, 1]  
  
x = points.count(9)

**Método list extend() de Python**

### **Example**

Add the elements of cars to the fruits list:

fruits = ['apple', 'banana', 'cherry']  
  
cars = ['Ford', 'BMW', 'Volvo']  
  
fruits.extend(cars)

**Definición y uso**

El método extend() agrega los elementos de la lista especificada (o cualquier iterable) al final de la lista actual.

**Sintaxis**

list.extend(iterable)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *iterable* | Requerido. Cualquier iterable (lista, conjunto, tupla, etc.) |

**Más ejemplos**

### **Example**

Add a tuple to the fruits list:

fruits = ['apple', 'banana', 'cherry']  
  
points = (1, 4, 5, 9)  
  
fruits.extend(points)

Actividades del 06-10 sept 2021

# **Python List** index()**Method**

### **Example**

What is the position of the value "cherry":

fruits = ['apple', 'banana', 'cherry']  
  
x = fruits.index("cherry")

**Definición y uso**

El método index() devuelve la posición en la primera aparición del valor especificado.

**Sintaxis**

list.index(elmnt)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| elmnt | Requerido. Cualquier tipo (cadena, número, lista, etc.).  El elemento a buscar. |

**Más ejemplos**

### **Example**

What is the position of the value 32:

fruits = [4, 55, 64, 32, 16, 32]  
  
x = fruits.index(32)

**Nota**: El método index() solo devuelve la primera aparición del valor.

Actividades del 15sept2021

**Método insert() de listas de Python**

### **Example**

Insert the value "orange" as the second element of the fruit list:

fruits = ['apple', 'banana', 'cherry']  
  
fruits.insert(1, "orange")

**Definición y uso**

El método insert() inserta el valor especificado en la posición especificada.

**Sintaxis**

list.insert(pos, elmnt)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| pos | Requerido. Un número que especifica en qué posición insertar el  valor. |
| elmnt | Requerido. Un elemento de cualquier tipo (cadena, número,  objeto, etc.) |

**Método Python List pop()**

### **Example**

Remove the second element of the fruit list:

fruits = ['apple', 'banana', 'cherry']  
  
fruits.pop(1)

**Definición y uso**

El método pop() elimina el elemento en la posición especificada.

**Sintaxis**

list.pop(pos)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| pos | Opcional. Un número que especifica la posición del elemento  que desea eliminar, el valor predeterminado es -1, que devuelve  el último elemento |

**Más ejemplos**

### **Example**

Return the removed element:

fruits = ['apple', 'banana', 'cherry']  
  
x = fruits.pop(1)

**Nota**: El método pop() devuelve el valor eliminado.

**Método remove () de Python List**

### **Example**

Remove the "banana" element of the fruit list:

fruits = ['apple', 'banana', 'cherry']  
  
fruits.remove("banana")

**Definición y uso**

El método remove() elimina la primera aparición del elemento con el valor especificado.

**Sintaxis**

list.remove(elmnt)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| elmnt | Requerido. Cualquier tipo (cadena, número, lista, etc.).  El elemento que desea eliminar. |

**Método inverse() de lista de Python**

### **Example**

Reverse the order of the fruit list:

fruits = ['apple', 'banana', 'cherry']  
  
fruits.reverse()

**Definición y uso**

El método reverse() invierte el orden de clasificación de los elementos.

**Sintaxis**

list.reverse()

**Valores paramétricos**

Sin parámetros

**Páginas relacionadas**

La función incorporada reversed() devuelve un objeto iterador invertido.

**Método Python List sort()**

### **Example**

Ordenar la lista alfabéticamente:

cars = ['Ford', 'BMW', 'Volvo']  
  
cars.sort()

**Definición y uso**

El método sort() ordena la lista de forma ascendente por defecto.

También puede crear una función para decidir los criterios de clasificación.

**Sintaxis**

list.sort(reverse=True|False, key=myFunc)

**Valores paramétricos**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| reverse | Opcional. reverse = True ordenará la lista de forma descendente.  El valor predeterminado es reverse = Falso. |
| key | Opcional. Una función para especificar los criterios de  clasificación. |

**Más ejemplos**

### **Example**

Ordenar la lista de forma descendente::

cars = ['Ford', 'BMW', 'Volvo']  
  
cars.sort(reverse=True)

### **Example**

Ordene la lista por la longitud de los valores::

# A function that returns the length of the value:

def myFunc(e):  
  return len(e)  
  
cars = ['Ford', 'Mitsubishi', 'BMW', 'VW']  
  
cars.sort(key=myFunc)

### **Example**

Sort a list of dictionaries based on the "year" value of the dictionaries:

# A function that returns the 'year' value:  
def myFunc(e):  
  return e['year']  
  
cars = [  
  {'car': 'Ford', 'year': 2005},  
  {'car': 'Mitsubishi', 'year': 2000},  
  {'car': 'BMW', 'year': 2019},  
  {'car': 'VW', 'year': 2011}  
]  
  
cars.sort(key=myFunc)

Print(cars)

### **Example**

Ordene la lista por la longitud de los valores e invierta:

# A function that returns the length of the value:  
def myFunc(e):  
  return len(e)  
  
cars = ['Ford', 'Mitsubishi', 'BMW', 'VW']  
  
cars.sort(reverse=True, key=myFunc)  
print(cars)