Hoja de trucos de estructuras de datos de Python

Lista

Paquete/Método	Descripción	Ejemplo de código

append()	El método `append()` se utiliza para agregar un elemento al final de una lista.	Sintaxis:
		`list_name.append(element)`
		Ejemplo:
		`fruits = ["manzana", "plátano", "naranja"]`
		`fruits.append("mango") print(fruits)`

copy()	El método `copy()` se utiliza para crear una copia superficial de una lista.	Ejemplo 1:
		`my_list = [1, 2, 3, 4, 5]`
		`new_list = my_list.copy() print(new_list)`
		`# Salida: [1, 2, 3, 4, 5]`

count()	El método `count()` se utiliza para contar el número de ocurrencias de un elemento específico en una lista en Python.	Ejemplo:
		`my_list = [1, 2, 2, 3, 4, 2, 5, 2]`
		`count = my_list.count(2) print(count)`
		`# Salida: 4`

Crear una lista	Una lista es un tipo de dato incorporado que representa una colección ordenada y mutable de elementos. Las listas están encerradas en corchetes [] y los elementos están separados por comas.	Ejemplo: `fruits = ["manzana", "plátano", "naranja", "mango"]`

del	La declaración `del` se utiliza para eliminar un elemento de la lista. La declaración `del` elimina el elemento en el índice especificado.	Ejemplo:
		`my_list = [10, 20, 30, 40, 50]`
		`del my_list[2] # Elimina el elemento en el índice 2 print(my_list)`
		`# Salida: [10, 20, 40, 50]`

extend()	El método `extend()` se utiliza para agregar múltiples elementos a una lista. Toma un iterable (como otra lista, tupla o cadena) y agrega cada elemento del iterable a la lista original.	Sintaxis:
		`list_name.extend(iterable)`
		Ejemplo:
		`fruits = ["manzana", "plátano", "naranja"]`
		`more_fruits = ["mango", "uva"]`
		`fruits.extend(more_fruits)`
		`print(fruits)`

Indexación	La indexación en una lista te permite acceder a elementos individuales por su posición. En Python, la indexación comienza en 0 para el primer elemento y llega hasta `length_of_list - 1`.	Ejemplo:
		`my_list = [10, 20, 30, 40, 50]`
		`print(my_list[0])`
		`# Salida: 10 (accediendo al primer elemento)`
		`print(my_list[-1])`
		`# Salida: 50 (accediendo al último elemento usando indexación negativa)`

insert()	El método `insert()` se utiliza para insertar un elemento.	Sintaxis:
		`list_name.insert(index, element)`
		Ejemplo:
		`my_list = [1, 2, 3, 4, 5]`
		`my_list.insert(2, 6)`
		`print(my_list)`

Modificar una lista	Puedes usar la indexación para modificar o asignar nuevos valores a elementos específicos en la lista.	Ejemplo:
		`my_list = [10, 20, 30, 40, 50]`
		`my_list[1] = 25 # Modificando el segundo elemento`
		`print(my_list)`
		`# Salida: [10, 25, 30, 40, 50]`

pop()	El método `pop()` es otra forma de eliminar un elemento de una lista en Python. Elimina y devuelve el elemento en el índice especificado. Si no proporcionas un índice al método `pop()`,	Ejemplo 1:
		`my_list = [10, 20, 30, 40, 50]`
		`removed_element = my_list.pop(2) # Elimina y devuelve el elemento en el índice 2`
		`print(removed_element)`
		`# Salida: 30`

	eliminará y devolverá el último elemento de la lista por defecto.	`print(my_list)`
		`# Salida: [10, 20, 40, 50]`
		
		Ejemplo 2:
		
		`my_list = [10, 20, 30, 40, 50]`
		`removed_element = my_list.pop() # Elimina y devuelve el último elemento`
		`print(removed_element)`
		`# Salida: 50`
		`print(my_list)`
		`# Salida: [10, 20, 30, 40]`

remove()	Para eliminar un elemento de una lista. El método `remove()` elimina la primera ocurrencia del valor especificado.	Ejemplo:
		
		`my_list = [10, 20, 30, 40, 50]`
		`my_list.remove(30) # Elimina el elemento 30`
		`print(my_list)`
		`# Salida: [10, 20, 40, 50]`

reverse()	El método `reverse()` se utiliza para invertir el orden de los elementos en una lista.	Ejemplo 1:
		
		`my_list = [1, 2, 3, 4, 5]`
		`my_list.reverse() print(my_list)`
		`# Salida: [5, 4, 3, 2, 1]`

Segmentación	Puedes usar la segmentación para acceder a un rango de elementos de una lista.	Sintaxis:
		
		`list_name[start:end:step]`
		
		Ejemplo:
		
		`my_list = [1, 2, 3, 4, 5]`
		`print(my_list[1:4])`
		`# Salida: [2, 3, 4] (elementos desde el índice 1 hasta 3)`
		`print(my_list[:3])`
		`# Salida: [1, 2, 3] (elementos desde el principio hasta el índice 2)`
		`print(my_list[2:])`
		`# Salida: [3, 4, 5] (elementos desde el índice 2 hasta el final)`
		`print(my_list[::2])`
		`# Salida: [1, 3, 5] (cada segundo elemento)`

sort()	El método `sort()` se utiliza para ordenar los elementos de una lista en orden ascendente. Si deseas ordenar la lista en orden descendente, puedes pasar el argumento `reverse=True` al método `sort()`.	Ejemplo 1:
		
		`my_list = [5, 2, 8, 1, 9]`
		`my_list.sort()`
		`print(my_list)`
		`# Salida: [1, 2, 5, 8, 9]`
		
		Ejemplo 2:
		
		`my_list = [5, 2, 8, 1, 9]`
		`my_list.sort(reverse=True)`
		`print(my_list)`
		`# Salida: [9, 8, 5, 2, 1]`

Tupla

Paquete/Método	Descripción	Ejemplo de código

count()	El método count() para una tupla se utiliza para contar cuántas veces aparece un elemento especificado en la tupla.	Sintaxis:
		
		`tuple.count(value)`
		
		Ejemplo:
		
		`fruits = ("manzana", "plátano", "manzana", "naranja")`
		`print(fruits.count("manzana")) #Cuenta cuántas veces se encuentra manzana en la tupla.`
		`#Salida: 2`

index()	El método index() en una tupla se utiliza para encontrar la primera ocurrencia de un valor especificado y devuelve su posición (índice). Si el valor no se encuentra, genera un ValueError.	Sintaxis:
		
		`tuple.index(value)`
		
		Ejemplo:
		
		`fruits = ("manzana", "plátano", "naranja")`
		`print(fruits[1]) #Devuelve el valor en el que está presente manzana.`
		`#Salida: plátano`

sum()	La función sum() en Python se puede usar para calcular la suma de todos los elementos en una tupla, siempre que los elementos sean numéricos (enteros o flotantes).	Sintaxis:
		`sum(tuple)`
		Ejemplo:
		`numbers = (10, 20, 5, 30)`
		`print(sum(numbers))`
		`#Salida: 65`

min() y max()	Encuentra el elemento más pequeño (min()) o más grande (max()) en una tupla.	Ejemplo:
		`numbers = (10, 20, 5, 30)`
		`print(min(numbers))`
		`#Salida: 5`
		`print(max(numbers))`
		`#Salida: 30`

len()	Obtén el número de elementos en la tupla usando len().	Sintaxis:
		`len(tuple)`
		Ejemplo:
		`fruits = ("manzana", "plátano", "naranja")`
		`print(len(fruits)) #Devuelve la longitud de la tupla.`
		`#Salida: 3`