Python Cheat Sheet: Tipos de datos complejos

	Descripción	Ejemplo
List	Un tipo de datos de contenedor que almacena una secuencia de elementos. A diferencia de las “string”, las listas son mutables: modificación posible.	`l = [1, 2, 2]` `print(len(l)) # 3`

Añadir elementos	Agregua elementos a una lista con (i) append, (ii) insert, o (iii) concatenación. La operación de adición es muy rápida.	`[1, 2, 2].append(4) # [1, 2, 2, 4]`
		`[1, 2, 4].insert(2,2) # [1, 2, 2, 4]`
		`[1, 2, 2] + [4] # [1, 2, 2, 4]`

Remover	Remover un elemento es más lento	`[1, 2, 2, 4].remove(1) # [2, 2, 4]`
Reverse	Da la lista en orden inverso	`[1, 2, 3].reverse() # [3, 2, 1]`
Ordenar	Sort() ordena la lista, de menor a mayor.	`[2, 4, 2].sort() # [2, 2, 4]`
Indexing	Encuentra la primera aparición de un elemento en la lista y devuelve su índice. Puede ser lento porque recorre toda la lista.	`[2, 2, 4].index(2) # index of element 4 is "0"` `[2, 2, 4].index(2,1) # index of element 2 after pos 1 is "1"`

Stack	Las listas de Python se pueden usar intuitivamente como pilas a través de las dos operaciones de lista append () y pop ().	`stack = [3]`
		`stack.append(42) # [3, 42]`
		`stack.pop() # 42 (stack: [3])`
		`stack.pop() # 3 (stack: [])`

Set	Un set es una colección desordenada de elementos únicos.	`basket = {'apple', 'eggs', 'banana', 'orange'}` `same = set(['apple', 'eggs', 'banana', 'orange'])`
Dictionary	El diccionario es una estructura de datos útil para almacenar pares (clave, valor).	`calories = {'apple' : 52, 'banana' : 89, 'choco' : 546}`

Leer y escribir elementos	Lee y escribe elementos especificando la clave entre corchetes. Utiliza las funciones keys () y values () para acceder a todas las claves y valores del diccionario.	`print(calories['apple'] < calories['choco']) # True`
		`calories['cappu'] = 74`
		`print(calories['banana'] < calories['cappu']) # False`
		`print('apple' in calories.keys()) # True`
		`print(52 in calories.values()) # True`

Bucles en diccionario	Puede acceder a los pares (clave, valor) de un diccionario con el método items ().	`for k, v in calories.items():` `  print(k) if v > 500 else None # 'chocolate'`

Operador IN	Compruebe con la palabra clave "in" si la lista, la lista o el diccionario contienen un elemento. La contención de conjuntos es más rápida que la contención de listas.	`basket = {'apple', 'eggs', 'banana', 'orange'}`
		`print('eggs' in basket) # True`
		`print('mushroom' in basket) # False`

List and Set Comprehension	La “comprehension” de listas es la forma concisa de Python para crear listas. Utiliza corchetes más una expresión, seguida de una cláusula for.
	Cierreacon cero o más cláusulas for o if. La “comprehension” en “set” es similar a la “comprehension” de “list”.
		`# List comprehension`
		`l = [('Hi ' + x) for x in ['Alice', 'Bob', 'Pete']]`
		`print(l) # ['Hi Alice', 'Hi Bob', 'Hi Pete']`
		`l2 = [x * y for x in range(3) for y in range(3) if x>y]`
		`print(l2) # [0, 0, 2]`
		`# Set comprehension`
		`squares = { x**2 for x in [0,2,4] if x < 4 } # {0, 4}`